CASINO

User's Guide Version 1.7.7

Richard Needs, Mike Towler, Neil Drummond, and Paul Kent

Theory of Condensed Matter Group
Cavendish Laboratory
Madingley Road
Cambridge CB3 0HE
UK

January 2005

Contents

1 Introduction **1**

2 The quantum Monte Carlo method **2**

3 Miscellaneous issues **3**
3.1 Support . . . 3
3.2 Legal agreements and being nice . . . 3
3.3 Getting the latest version of the code . . . 3

4 Functionality of CASINO **4**

5 Installation **5**
5.1 Supported architecture . . . 5
5.2 Unsupported architecture . . . 6
5.3 Further installation notes . . . 6

6 How to use CASINO **7**
6.1 Getting started : summary . . . 7
6.2 Getting started : details . . . 7
6.3 How to run the code : run scripts . . . 18

7 Files used by CASINO **21**
7.1 Basic input file : input . . . 22
7.2 Jastrow factor file: `jastrow.data` . . . 32
7.3 Pseudopotential file : `xx_pp.data` . . . 37
7.4 Charge density file : density.data . . . 38
7.5 MPC file : eepot.data . . . 39
7.6 Trial wave function files : awfn.data, bwfn.data, gwfn.data, pwfn.data, swfn.data . . . 40
7.7 heg.data file . . . 49
7.8 Jellium slabs : heg.data file with ETYPE=6. . . . 52
7.9 vmc.hist file . . . 54
7.10 dmc.hist and dmc.hist2 files . . . 55

8 Specifying the Slater determinants **56**
8.1 Gaussian basis sets . . . 56
8.2 Plane wave basis . . . 57
8.3 Blip and spline basis . . . 60

9 Pseudocode **61**
9.1 Subroutine VMC . . . 61
9.2 Subroutine DMC . . . 61

10 Theoretical background **62**
10.1 The trial wave function . . . 62
10.2 The variational Monte Carlo method . . . 62
10.3 The diffusion Monte Carlo method . . . 64
10.4 Drift and diffusion . . . 65
10.5 Branching and population control . . . 67
10.6 Modifications to the Green's Function . . . 68
10.7 Modifications to the DMC Green's function at bare nuclei . . . 71
10.8 Evaluating expectation values of observables . . . 72
10.9 Growth estimator of the energy . . . 73
10.10 Automatic block-resetting . . . 74
10.11 Evaluation of orbitals in the determinant part of the wave function . . . 74
10.12 Constructing real orbitals . . . 75
10.13 Cusp corrections for Gaussian orbitals . . . 75

10.14The Jastrow factor . . . 79
10.15Wave function updating . . . 82
10.16Evaluating the local energy . . . 83
10.17Evaluating the kinetic energy . . . 84
10.18Evaluating the non-local pseudopotential energy . . . 85
10.19The core polarization potential energy . . . 85
10.20Evaluation of infinite Coulomb sums . . . 87
10.21Estimating equilibration times and correlation periods . . . 91
10.22Statistical analysis of data . . . 92
10.23Wave function optimization - standard method . . . 94
10.24Wave function optimization - new method . . . 96
10.25Further considerations in electron-hole systems . . . 97
10.26Pair correlation function calculation for electron/electron-hole systems . . . 99
10.27Relativistic corrections to atomic energies . . . 100

11 Making movies with CASINO **102**
11.1 How to make movies . . . 102
11.2 Visualisation . . . 102

12 Using CASINO with the CRYSTAL program **104**
12.1 The CRYSTAL program . . . 104
12.2 Generating `gwfn.data` files with CRYSTAL95/98 and *crystaltoqmc* . . . 104
12.3 Generating gwfn.data files with CRYSTAL03 . . . 105

13 Using CASINO with the Gaussian94/98/03 programs **106**
13.1 How to use *gaussiantoqmc* . . . 106
13.2 Other features of *gaussiantoqmc* . . . 107
13.3 Summary of routines used in *gaussiantoqmc* . . . 110

14 Use of localized orbitals and bases in CASINO **112**
14.1 Theoretical background . . . 112
14.2 Using CASINO to carry out "linear-scaling" QMC calculations . . . 113

15 Using CASINO with Blip functions **116**
15.1 Blip functions . . . 116
15.2 The blip conversion utility . . . 116
15.3 The blipl conversion utility . . . 116

16 Using CASINO with other supported programs **117**
16.1 TURBOMOLE . . . 117
16.2 CASTEP 2002 . . . 117
16.3 PWSCF . . . 117
16.4 ABINIT . . . 117

17 Using CASINO with unsupported programs **117**

18 Utilities provided with the CASINO distribution **118**

19 Appendix 1 : Old Jastrow factor file: jasfun.data **121**

20 Appendix 2 : Programming Guide for CASINO **123**
20.1 STYLE . . . 123
20.2 CONTENT . . . 125
20.3 PERFORMANCE . . . 126
20.4 BUG REPORTS . . . 127
20.5 REQUESTS FOR NEW FEATURES . . . 127

Bibliography **128**

1 Introduction

CASINO is a code for performing quantum Monte Carlo (QMC) electronic structure calculations for finite and periodic systems. Its development was inspired by a Fortran77 development code (known simply as 'the QMC code') written in the early 1990s in Cambridge by Richard Needs and Guna Rajagopal, assisted by many helpful discussions with Matthew Foulkes. This was later extended by Andrew Williamson up to 1995 and then by Paul Kent and Mike Towler up to 1998. Various different versions of this were able to treat fcc solids, single atoms and the homogeneous electron gas. By the late 1990s it was clear that a modern general code capable of treating arbitrary systems (e.g. at least atoms, molecules, polymers, slabs, crystals, and electron phases) was required, not only for the use of the Cambridge QMC group, but for public distribution. At that time, a user-friendly general publically available code did not exist (at least for periodic systems), and it was felt to be a good thing to create one to allow people around the world to join in the fun.

So beginning in 1999 a new Fortran90 code, CASINO, was gradually developed in the group of Richard Needs largely by Mike Towler, considerably assisted from 2002 by tall Ph.D. student Neil Drummond (some routines from the old code were retained, translated and reused, although most were gradually replaced). The main aims of the new code were generality, speed, ease-of-use and transferability over a wide range of computational hardware. It is hoped that these objectives have been largely attained, but the code continues to be actively developed.

Over the years, additional contributions have been made by Andrew Porter, Randy Hood, Andrew Williamson, Dario Alfè, Gavin Brown, Chris Pickard, Rene Gaudoin, Ben Wood, Zoltan Radnai, Andrea Ma, Pablo Lopez Rios, Ryo Maezono, John Trail and possibly others, for which we are grateful.

Finally, note that users need to sign the relevant piece of paper (available from Mike Towler or Richard Needs) before using the code, and that the following citation (quoted in full) is required in any publication describing results obtained with CASINO:

R. J. Needs, M. D. Towler, N. D. Drummond and P. R. C. Kent, CASINO version 1.7 User Manual, University of Cambridge, Cambridge (2004).

Further public information and resources are available at the CASINO web page:

www.tcm.phy.cam.ac.uk/∼mdt26/casino.html

The CASINO pseudopotential library and download facility can be found on the private CASINO users page:

www.tcm.phy.cam.ac.uk/∼mdt26/casino_users.html

2 The quantum Monte Carlo method

The correlated motion of electrons plays a crucial role in the aggregation of atoms into molecules and solids, in electronic transport properties and in many other important physical phenomena. *Ab initio* electronic structure calculations in which the properties of such correlated electron systems are computed from first principles are a vital tool in modern condensed matter physics and molecular quantum chemistry. Currently the most popular way to include the effects of electron correlation in these calculations is *density functional theory*. This method is in principle exact, in reality fast and often very accurate, but does have a certain number of well-known limitations. In particular, with only limited knowledge available concerning the exact mathematical form of the so-called exchange-correlation functional, the accuracy of the approximate form of the theory is non-uniform and non-universal, and there are important classes of materials for which it gives qualitatively wrong answers.

An important and complementary alternative for situations where accuracy is paramount is the *quantum Monte Carlo* (QMC) method, which has many attractive features for probing the electronic structure of real systems. It is an explicitly many-body method applicable to both finite and periodic systems which takes electron correlation into account from the outset. It gives consistent, highly accurate results while at the same time exhibiting favourable scaling of computational cost with system size. This is in sharp contrast to the accurate methods used in mainstream quantum chemistry such as configuration interaction or coupled cluster theory which are impractical for anything other than small molecules and cannot generally be applied to condensed matter problems.

The use of quantum Monte Carlo has been greatly hampered over the last two decades by a combination of insufficient computer power and a lack of available, efficient QMC computer programs general enough to treat a similar range of problems to regular DFT codes. Fast parallel computers are now widespread, and you are now in possession of just such a computer program that is capable of carrying out QMC calculations for a wide range of interesting chemical and physical problems on a variety of computational hardware. CASINO has been designed to make the power of the quantum Monte Carlo method available to everyone, and we hope you enjoy using it.

3 Miscellaneous issues

3.1 Support

CASINO has documentation and examples etc. but it is a research code and learning how to use it is a significant task. This is particularly the case if the user does not have relevant experience such as familiarity with VMC and DMC calculations and knowledge of DFT/HF methods for atoms, molecules and solids. We are finding that supporting users can take a large amount of our time and so we are having to limit the number of such groups that we can work with directly. We are finding that most people need quite a lot of help and the project turns into a collaboration, but of course we cannot enter into too many projects of this type as our time is limited. We do have people visiting here to learn about the codes and how to do calculations, and this seems to work well.

In general, you are welcome to have a copy of CASINO if you sign and adhere to our agreement, and we can give you some support in using it, but please understand that this may well be rather limited in extent.

Having noted the above feel free to mail Richard Needs (rn11 at cam.ac.uk) or Mike Towler (mdt26 at cam.ac.uk) with any questions you may have. For more general questions you can use our mailing list (casino_users at phy.cam.ac.uk) .

3.2 Legal agreements and being nice

We are making CASINO available to a number of groups now. We are not currently asking for any payment for academic use of the code, but we ask users to sign an agreement concerning use of CASINO. The practical upshot of this is that users may not redistribute the code, they may not incorporate any part of it into any other program system, nor may they modify it in any way whatsoever *without prior agreement of the Cambridge group.* Please read this agreement carefully and abide by what it says. You may not use or even retain a copy of CASINO if you have not signed this agreement. A copy of the agreement can be found in the CASINO distribution in file `CASINO/docs/consent_form.ps`.

New versions of CASINO are produced on a regular basis (nightly builds for the beta version). As a courtesy to the authors it would also be appreciated if users refrain from publishing scientific articles using recently introduced facilities in the code if the authors of those facilities have not yet published their work (at least, without first checking with them). Thanks for your understanding on this point.

It would be greatly appreciated if you could forward a copy of any article published using the results of CASINO calculations to us, both for our interest and so we can add references to the CASINO web pages.

3.3 Getting the latest version of the code

Registered users can download recent versions of CASINO from our secure web site : `www.tcm.phy.cam.ac.uk/~mdt26/casino_users.html`. It should ask you for a username and a password. These are normally assigned to particular institutions - if you don't have one then please contact Mike Towler who will be pleased to help.

The site will normally contain the most recent 'stable release' and a 'beta version'. The latter is a nightly build released most evenings with the latest changes. Clearly it isn't guaranteed not to fall over and you use it at your own risk. To help you decide which version you need, a link to the CASINO `DIARY` file is provided which contains a comprehensive list of all modifications to the code.

Users in the Cambridge TCM group who belong to the Unix group 'casino' may additionally copy the latest source from the directory `~casino/current_beta`.

4 Functionality of CASINO

The CASINO program is actively being developed and many improvements and revision are envisaged, but in its present state its capabilities are as follows:

- Variational Monte Carlo (including variance minimization of wave functions).
- Diffusion Monte Carlo (branching DMC and pure DMC).
- Use of Slater-Jastrow wave functions where the Slater part may consist of multiple determinants of spin orbitals.
- Trial wave functions expanded in Gaussian basis sets (s, sp, p, d, f or g functions centred on atoms or elsewhere) produced by the following programs (using DFT, HF, or various multideterminant methods): CRYSTAL9X/03 [24] [25], GAUSSIAN9X/03 [26] [27] and TURBOMOLE.
- Trial wave functions expanded in plane waves generated from PWSCF [30], ABINIT [29], GP, CASTEP [28] and k207.
- Trial wave functions expanded in blip functions generated by post-processing plane-wave DFT solutions.
- Trial wave functions in spline functions generated by post-processing plane-wave DFT solutions.
- Numerical atomic calculations with the orbitals interpolated from a radial grid.
- Improved scaling with system size through use of Wannier orbitals and localized basis functions.
- Computation of excitation energies corresponding to either promotion or addition/subtraction of electrons.
- Computation of distribution functions such as the pair correlation function and density matrices (electron and electron-hole systems only for the moment).
- 2D/3D electron phases in fluid or crystal wave functions, with arbitrary cell shape/spin polarization/density (including excited state capability). 2D layer separation between opposite spin electrons possible. Pair correlation function.
- 2D/3D electron-hole phases with fluid/crystal/pairing wave functions with arbitrary cell shape/spin polarization/density (including excited state capability). Variable electron-hole mass ratio. 2D layer separation between holes and electrons possible. Pair correlation function(s).
- Calculation of electron-electron interactions using either Ewald and/or our 'modified periodic Coulomb interaction' which is faster and has smaller Coulomb finite size effects.
- Parallelized using MPI—tested in parallel on Hitachi SR2201 and SR11000, Cray T3E, SGI Origin 2000, SGI Altix, IBM SP3 and IBM Pseries, Fujitsu Primepower, Alpha servers and SunFire Galaxy, Linux PC clusters, Opteron clusters.
- Also set up for workstation use on DEC Alphas, SGI Octane and O2, Linux PC with various compilers. MPI libraries not required on single processor machines.
- Keyword input handled using highly flexible ESDF electronic structure data format.
- Self-documenting help system.

5 Installation

Unpack the CASINO_vxxx.tar.gz distribution in your *home* directory, then go to 5.1 or 5.2 depending on whether your architecture is supported or not.

5.1 Supported architecture

This currently includes (at least):
Workstations: DEC Alpha, SGI Octane and O2, Linux PC (ifc, pgf90, nag, compaq, lahey lf95 compilers)
Parallel Machines: Hitachi SR2201, Cray T3E, SGI Origin 2000, SGI Altix, IBM SP3, IBM P-series, Fujitsu Primepower, SunFire Galaxy, Alpha multiprocessors, Linux PC clusters, Linux PC multiprocessors, Opteron clusters

See the file CASINO/ARCH for the definitive list of supported architectures with definitions and library requirements.

The instructions below require you to set environment variables. The procedure for doing this depends on what Unix shell you use. Roughly speaking, if you use the (t)csh shell, then environment variables should be set in your .cshrc file using commands like :

```
setenv QMC_ARCH sun
setenv QMC_TMPDIR /temp/$user
set path = ($path $HOME/CASINO_bin_qmc/utils/$QMC_ARCH)
```

If you use the bash shell, then add something like the following to your .bashrc file.

```
export QMC_ARCH="sun"
export QMC_TMPDIR="/temp/$USER"
export PATH=$PATH:$HOME/CASINO/bin_qmc/utils/$QMC_ARCH
```

After modifying the appropriate file, you should type 'source /.cshrc' or 'source /.bashrc'. (Make sure these files are read automatically during the login procedure that you use - if not, include the 'source' line above in your .login or .bash_profile file.) If you use a shell other than these three, then you're on your own..!

Set the environment variable QMC_ARCH to be *machine_type*, where *machine_type* is one of the supported architectures listed in the CASINO/ARCH file. This variable identifies what kind of computer you are running on - so that CASINO knows what Fortran compiler to use, what libraries are required, how jobs are run, and so on. If you are not sure which one you should use, take a look at the list in this file, and/or the files in the CASINO/src/zmakes/ directory.

2. Add /CASINO/bin_qmc/utils/$QMC_ARCH to your path (using shell-dependent commands - see above for examples).

3 Type *make* in the ∼/CASINO/src directory.

4 Type *make* in the ∼/CASINO/utils directory.

5. When compilation has finished, type *rehash* (if using csh or tcsh). All required programs will now be in your path.

6. That's it.

If e.g. the default location of the MPI library doesn't work on your machine, you can personalize it by setting an environment variable QMC_ID = [your machine name]. Create a file called CASINO/src/zmakes/users/$QMC_ARCH/$QMC_ID.inc containing override definitions for the

library locations. If you send such files to Mike (mdt26 at cam.ac.uk) then they can be permanently included in the CASINO distribution if you wish. The QMC_ID variable need not be set if you are happy with the defaults. Active QMC_IDs are listed at the bottom of the CASINO/ARCH file.

Note that the MPI library is not required to compile CASINO on single-processor machines. No other external libraries are required.

5.2 Unsupported architecture

1. Write a CASINO/src/zmakes/[machine_name].inc file containing compiler name and appropriate flags for optimized/debugged/profiled code using the existing include files as a model. Copy it (with the evident modifications) into CASINO/utils/zmakes/[machine_name].inc. If your machine uses non-standard library locations or compiler flags or whatever you may also create a CASINO/src/zmakes/[machine_type]/[machine_name].inc file containing personalized settings which can be accessed through the QMC_ID environment variable (see note about this in 'Supported architectures' above.).

2. Now you need some run scripts to run the code.

If your machine is a workstation/multiprocessor PC/cluster which does not use a batch queue system, then you can just use the standard *runvmc*, *runvarmin* and *rundmc* from the CASINO/scripts/workstation directory. You will need to add a few lines to the CASINO/utils/Makefile so that these scripts are copied from this directory if QMC_ARCH is set to the new machine type.

If your machine is a parallel machine using a batch queue system, then you will probably need to adapt existing run scripts. Obviously you can use the ones in the origin/t3e/hitachi/ibm_sp3 etc. directories as models. Running CASINO without these scripts is possible but not really recommended, since the scripts have been carefully tuned to check for most common errors, to run CASINO in as efficient a way as possible, and to tidy up after it.

3. You may also need to do a little further hacking around in the CASINO/utils/Makefile in order to get all the utilities working properly. Note that many of these utilities require you to be able run interactive jobs which may not be impossible on certain batch-based machines, so make sure the Makefile doesn't bother to compile and set them up. Have a look at the CASINO/utils/help/casinohelp script too, to make sure it behaves appropriately with your new machine type.

4. Send 1, 2 and 3 to Mike Towler at mdt26 at cam.ac.uk

5. Go to supported architecture list above.

5.3 Further installation notes

- By default all Fortran90 files will be compiled with full optimization. If you want to compile with debugging or profiling compiler options instead of the optimizing ones, or to compile a separate 'development version', then type *make debug*, *make prof* or *make dev*. The Makefile will keep separate copies of all object files and binaries for the different compilation levels.
- Note that on a multi-machine environment like the Cambridge High Performance Computing Centre where many different machines access the same filespace you can use the same CASINO source distribution for all machines. The Makefile keeps separate compilation records for each machine and source type (`opt/debug/prof/dev`)—thus you can compile the same source on more than one different machine at the same time.

6 How to use CASINO

6.1 Getting started : summary

You should first learn how to use CASINO using the examples provided. Example input files for finite systems (atoms, molecules) and for systems with periodicity in one, two or three dimensions (polymers, slabs and crystalline solids) plus various electron and electron-hole systems can be found in the /CASINO/examples directory. If you have set up the code as described in the installation section, you can run the code with the *runvmc*, *rundmc* and *runvarmin* scripts which will have been placed in your path. Change the units and reblock the answers using the utility *reblock*. The behaviour of the QMC calculation is determined by a list of keywords in the file `input`. To access the internal CASINO help system which tells you about this, type *casinohelp all* to get a list of all keywords that the program knows about, or *casinohelp keyword* for detailed help on a particular keyword. Create a blank file containing the keyword **input_example** and type *runvmc*—this will create a sample input file containing all valid input keywords and their default values in the correct format.

6.2 Getting started : details

This section gives basic practical details for running QMC calculations with CASINO and is intended for new users. It assumes you already know something about the theory of VMC, DMC and variance minimization. If you don't, then please see the standard references for general details about the methods, and the Theoretical Background section in this manual for details about how the algorithms are implemented in CASINO. Following the instructions in this section will not necessarily lead to publication quality results, but should at least allow you to play with the code and get some feel for how it works.

6.2.1 Trial wave functions and pseudopotentials

Unless you're interested in electron and electron-hole phases in the absence of an external potential (in which case you can start straight away) the first hurdle to doing research with CASINO is to generate a trial wave function using, for example, a DFT or Hartree-Fock calculation (multideterminant quantum chemistry methods can also be used). This must be done using an external program, which must in the past have been persuaded to write out the wave function in a format that CASINO understands, either all by itself, or through the transformation of the standard output of the program using a separate utility (which will probably be included in the directory `CASINO/utils/wfn_converters`). Note that 'writing out the wave function' basically means writing out the geometry, the basis set, and the orbital coefficients.

The information defining the trial wave function generated by the external program lives in files whose name depends on the basis set in which the orbitals in the determinantal part of the wave function are expanded. They are called `gwfn.data` (Gaussians), `pwfn.data` (plane waves), `bwfn.data` (blip functions), `swfn.data` (splines) or `awfn.data` (atomic orbitals given explicitly on a radial grid). These files will often be referred to generically with the name `xwfn.data`. Choice of basis set has been found to depend largely on personal prejudice, though some consideration should be given to issues of computational efficiency.

Gaussian and plane-wave functions are generated directly. Blip and spline wave functions are generated by post-processing a plane-wave `pwfn.data` file using a CASINO utility (this is desirable since plane waves are the absolute worst basis set you can choose for QMC - every basis function contributes at every point in space - adding a factor of N to the scaling with system size). Splines and blips are essentially the same thing written by two different groups. However, splines have now been flagged as obsolescent and users of this functionality should move over to using blip functions; the splines will be removed in a future version of CASINO.

To make the CASINO calculation scale independently of system size (energy per atom properties) or quadratically (energy per cell properties) then the delocalized orbitals generated by most DFT/HF programs need to undergo a unitary transformation to localized (Wannier) form. This can be

done (for plane waves orbitals only at the moment) either by using the CASINO *xwannier* utility (splines) or *localizer* utility (blips). The orbitals should then be re-expanded in splines (using the *generate_spline* utility), or blips (using the blipl utility).

Generating trial wave functions currently requires you to have access to one of the following codes. This list may increase in the future - contact Mike Towler if you want your code to be supported.

Gaussian basis sets:

CRYSTAL95/98/03 : www.tcm.phy.cam.ac.uk/∼mdt26/crystal.html and links from there.

GAUSSIAN94/98/03 : www.gaussian.com

TURBOMOLE : www.chemie.uni-karlsruhe.de/PC/TheoChem/turbomole/intro.en.html.

Plane wave basis sets:

ABINIT : www.abinit.org. See reference [29]. Contact Mike Towler - mdt26 at cam.ac.uk - to discuss the current status of the interface.

CASTEP : The modern CASTEP [28] is an entirely new Fortran90 version of the venerable Cambridge plane-wave program which it is designed to replace. It is distributed by Accelrys - see www.accelrys.com/mstudio/castep.html. Mail Chris Pickard (cjp20@phy.cam.ac.uk) to discuss CASTEP and how to get hold of a copy.

k207 : A basic plane-wave program apparently only used by Richard Needs.

GP : If you don't work at Lawrence Livermore you're not allowed to use this, as it is a Government Secret. If you do, then obviously you already know about it.

PWSCF : www.pwscf.org. See reference [30]

Atoms using a radial grid:

TCM atomic code : mail Richard Needs rn11 at cam.ac.uk

Pseudopotentials

CASINO is capable of running all-electron calculations, where core and valence electrons are explicitly included in the simulation, or pseudopotential calculations, where the core electrons are replaced by an effective potential. The latter is normally advantageous since the computer time required for all-electron calculations scales rather badly with atomic number Z; this scaling is improved by using pseudopotentials. However, recent improvements to the algorithms used by CASINO mean that all-electron calculations may be performed for systems containing much heavier atoms than is normally believed to be sensible (test calculations up to Z=54 have been done here in Cambridge).

CASINO reads pseudopotential data from a file called `xx_pp.data` where `xx` is the chemical symbol of the element in lower case letters. The file contains a logarithmic radial grid and the values of the different angular momentum components of the pseudopotential at each grid point. A library of pseuopotentials suitable for use with CASINO is available at :

www.tcm.phy.cam.ac.uk/∼mdt26/casino_users.html

An important point is that exactly the same pseudopotential should be used in the DFT/HF calculation that generates the trial wave function and in CASINO. Other programs do not in

general understand the CASINO pseudopotential format, and so the information must somehow be transformed so that they do.

For programs using Gaussian basis sets such as GAUSSIAN9X and CRYSTAL, the pseudopotential must be reexpanded in Gaussian functions times powers of r. If you are using the Cambridge pseudopotentials, these expansions (in formats suitable for these two programs) are included in the on-line library. If you are using some other pseudopotential, then there is a CASINO utility - *ppfit* - which can be persuaded to do the expansion for you. There is an additional utility - *ptm* - which can manipulate pseudopotential files on grids and their Gaussian expansions in various useful ways.

As for plane wave programs, CASTEP understands the CASINO grid format and can read such files directly. Other plane wave programs require conversion utilities, which may be included in the CASINO/utils/pseudo_converters directory.

For atomic calculations on radial grids, `awfn.data` files generated with the Cambridge pseudopotentials are available in the on-line library.

6.2.2 Trial wave functions : emergencies

If you don't have access to any of these codes and have a specific system in mind, then Mike Towler is known for being able to generate trial wave functions in record time, on payment of a suitable fee (just to get you started he likes, in no particular order: bright shiny things, books about romance, yummy things to eat, poems, authorship on papers for which he hasn't really done any work, cute cuddly toys especially bears, and money).

6.2.3 Input file

Having prepared a trial wave function file and (perhaps) a pseudopotential file, you need to tell CASINO exactly what to do with them. CASINO takes its instructions from a file called `input` which contains a flexible list of keywords which control the behaviour of the calculation, switch on and off various options, and so on. Take a moment to examine the various `input` files in CASINO/examples/ to get a feel for what they look like.

A complete list of input keywords, together with their definitions, is contained in section 7.1. Further details, including default values, may be found by using the *casinohelp* utility. This tends to be more up to date than the manual, since it interrogates CASINO directly. Type *casinohelp all* to get a list of all keywords that CASINO knows about, or *casinohelp keyword* for detailed help on a particular keyword. Type *casinohelp search text* to search for the string *text* in all the keyword descriptions. Create a blank file containing the keyword **input_example** and type *runvmc*—this will create a sample input file containing all valid input keywords and their default values in the correct format.

Although there are many keywords, the beginning user can play around by changing only a few of them. Here's a (very) rough guide. Advice on good values to use will be given in the subsequent sections explaining how to do VMC, DMC and variance minimization calculations.

General (system dependent) keywords you should always change (in addition to the title given in the `input` file).

neu/ned : number of electrons of up and down spin
isperiodic : whether the system is periodic or not
npcells : number of primitive cells making up the simulation cell (not required for finite systems)

Important VMC keywords :

nequil : number of Metropolis equilibration steps
nmove : number of VMC moves
dtvmc : VMC time step

corper : VMC correlation period

Important DMC keywords :

nmove : number of VMC moves to perform in the config generation step
nwrcon : number of configs per node to generate in the config generation step
nconfig : number of configs per node to use in DMC.
nmove_dmc_equil : number of moves to use in DMC equilibration.
nblock_dmc_equil : number of blocks to use in DMC equilibration.
nmove_dmc_stats : number of moves to use in DMC statistics accumulation.
nblock_dmc_stats : number of moves to use in DMC stats accumulation.
dtdmc : DMC time step

Important variance minimization keywords :

nmove : number of VMC moves to perform in the config generation step
nwrcon : number of configs per node to generate in the config generation step
corper : VMC correlation period

6.2.4 Jastrow factor file

If you run a CASINO VMC calculation using a trial wave function consisting only of a single determinant of orbitals, then the result will be the Hartree-Fock energy. If the orbitals were generated using a Hartree-Fock calculation, then the VMC-HF energy should agree with the HF energy from the other code. This is a good check that everything is being done correctly. Obviously if the determinant is made up of Kohn-Sham orbitals from a DFT calculation, then the energies will not agree since the DFT program adds an exchange-correlation energy deduced from the self-consistent density.

The full Slater-Jastrow trial function normally used in QMC requires the determinantal part of the wave function stored in `xwfn.data` to be multiplied by a separate 'Jastrow factor' which defines the functional form of explicit interparticle correlations. In a typical VMC calculation, one might recover 60-80 per cent of the correlation energy in this way. This is not really enough to be generally useful, and in practice the main use of VMC is to prepare an accurate trial wave function given as input to a diffusion Monte Carlo (DMC) calculation. The DMC energy does not in principle depend on the Jastrow factor since it is positive definite and the DMC energy depends only on the nodal surface (the set of points in configuration space where the many-electron wave function is zero) . However, it makes the calculation vastly more efficient, and in general Jastrow factors should always be used.

The Jastrow factor is stored in a file called `jastrow.data`. Again, you should look at the examples to see what these look like. The various parameters in the files are defined in section 7.2. The adjustable parameters in the file must be optimized for a specific system, and this is the purpose of the variance minimization procedure.

6.2.5 Other files

If for a two- or three-dimensionally periodic system you want to use the *modified periodic Coulomb (MPC) interaction* to calculate the electron-electron energies instead of (or as well as) the standard Ewald interaction, then you need to generate two more files for the given geometry before you start doing VMC/DMC calculations. These are `eepot.data` (containing the Fourier transform of the $1/r$ interaction, and `density.data` (containing the Fourier transform of the charge density). These should be prepared for a given trial wave function and geometry by setting the input keyword **irun** to 5 and 6 respectively, and typing *runvmc*.

One might consider using the MPC interaction because CASINO can evaluate it much more quickly than the Ewald interaction, and also because it gives rise to smaller finite size effects. More details are given in section 10.20.4.

6.2.6 How to do a VMC calculation

Let's begin by calculating the Hartree-Fock energy of a hydrogen atom. Change directory to `CASINO/examples/atom/hydrogen`. You will see a `gwfn.data` file generated by a GAUSSIAN94 calculation (see the bottom of `gwfn.data` for the GAUSSIAN output) and a CASINO `input` file. No pseudopotential file is supplied, so CASINO will assume you wish to do an all-electron calculation. No `jastrow.data` file is supplied, because the correlation energy in a one-electron atom (zero!) is not difficult to calculate without one.

Look in the `input` file. You will see that **neu** and **ned** have been given the correct values (one spin-up electron present), **isperiodic** is false, and as this is a finite system the **npcells** block is not required. It is not necessary to equilibrate the electron distribution since there is only one electron, but **nequil** is set to 1000 to remind you that this should normally be done. The **nmove** parameter is set to 100000. The **corper** parameter is set to 3 to reduce serial correlation.

Type *runvmc*. Three files `out`, `vmc.hist` and `vmc.posout` should be produced. First type *ve*. This is a quick way to pull the VMC result out of the output file. It says :

```
Total energy :  -0.500009481495 +/- 0.000046640057 Time :  2.9000 seconds
```

Note that because of the stochastic nature of the method, the energy has an associated error bar (users of DFT etc. may find this disconcerting!). The true Hartree-Fock energy of the hydrogen atom is exactly -0.5 a.u., so we have seemingly correctly performed out first VMC calculation!

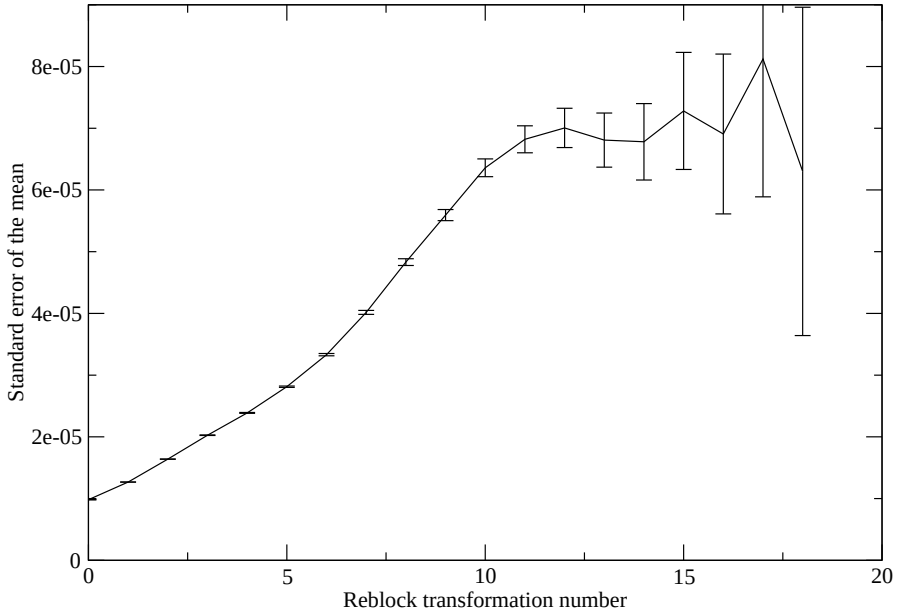
Next look in the file `out`. We see a complete report of the calculation, and at the end we see the total energy and its components. Just before the total energy, we see the 'acceptance ratio'. This should be around 50%. If it isn't, then you should play with the VMC timestep **dtvmc** (either manually, or by activating the **opt_dtvmc** flag with **nequil** set to at least 500 whereupon the timestep will be optimized automatically). If the acceptance ratio is very large, then the electrons are only attempting to move very short distances at each step of the random walk. They are thus likely to be bumbling about in one corner of the configuration of space and the serial correlation is likely to be large - increase **dtvmc** to prevent this. If the acceptance ratio is small, then the electrons are attempting very large jumps which are likely to be to regions of lower probability - and are thus likely to be rejected by the Metropolis algorithm. Thus the electrons don't move very much, and again the serial correlation is large. Reduce **dtvmc** to prevent this.

Next look at the file `vmc.hist` (see section 7.9). This contains mainly energy data produced at each step of the random walk (you can average over n successive steps using the **nvmcave** keyword to prevent this file getting too large). The `vmc.hist` file contains all the information necessary to perform a statistical analysis of the data.

If you want an accurate estimate of the error bar (you should!) then you must analyze the data in vmc.hist using the utility *reblock* which analyzes the serial correlation (this utility will also let you change the units of the answer). Type *reblock* in the directory containing the `vmc.hist` file. It will first ask you what units you want. Then it will print out a reblocking analysis (see section 10.22). The error bar will generally be (too) small for small values of the block length, and then should hopefully go up to a roughly constant value for higher block lengths, and for very high block lengths it will oscillate as the error bar on the error bar is very high. This constant value in the middle is the accurate error bar you want. Note that in general VMC calculations will reach this plateau more quickly than DMC calculations since they generally use a much larger time step and there is thus less serial correlation.

A file `reblock.plot` is normally produced as part of the output of the reblock procedure. If you have the programs *xmgr* or *xmgrace* set up on your system, then you can visualize the results of reblock by typing *plot_reblock* - you should be able to see a 'plateau' in the variance versus block length curve like in the figure below. Note that *xmgrace* is a very useful thing to have and it is used by various CASINO utilities. If you don't have it set up on your system

then you can download it for free from the following website : `plasma-gate.weizmann.ac.il/Grace/`



The file `vmc.posout` contains the final positions of the electrons and the current state of the random number generator so that the VMC run may be continued if desired. For example, if the error bar is still too large after a certain number of moves then you can make it smaller by running the simulation for longer. To do this, set the input keyword **inew** to 0, and rename the `vmc.posout` file to `vmc.posin` (in fact the *runmvc* script will normally ask if it can do this for you) then run the calculation for another **nmove** moves. All the extra data will be put onto the end of `vmc.hist` and the error bar from the reblocking analysis will be smaller and can in principle be made as small as desired.

Finally, go and find a system with more than one electron and a Jastrow factor such as the beryllium dimer in `molecules/be2` in the examples directory. Experiment with switching **irun** between 1 (VMC-HF) and 2 (VMC with Jastrow) and see the energy and error bar lowering effect of the Jastrow factor.

without Jastrow :

```
Total energy :  -29.102495953655 +/- 0.012473252263 Time :  9.3100 seconds
```

with Jastrow :

```
Total energy :  -29.221776570372 +/- 0.002571501258 Time :  12.2700 seconds
```

After any CASINO calculation, you can normally delete all the output that CASINO produces and return the directory back to the state it was in at the start of the calculation by typing *clearup*.

6.2.7 How to do a variance minimization calculation

[NOTE : there is now a new method of optimization used in CASINO for linear parameters which is about thirty times faster than the old one. It is not yet documented in this introductory section - but see section 10.24.]

The values for the parameters in the Jastrow factor used in the last part of the previous section (and other adjustable parameters such as coefficients in a multideterminant expansion) are optimized through a variance minimization calculation. This is probably the most difficult part of QMC for beginners, and perseverance will help. The user might first like to take a look at section 10.23, which contains a detailed summary of the theory and best practice. Here we simply summarize.

We wish to minimize the variance, the value of which is given by the first integral expression in section 10.23. This integral is approximated by summing over a fixed set of 'configurations' (i.e. values of the vector R giving the positions of all the electrons, and associated energies). These configurations must clearly be distributed according to the square of the trial wave function, and the first part of a variance minimization calculation is to generate them. This is done through an ordinary VMC run with the keyword **irun** set to 2, and **nwrcon** set to the number of configurations to generate (clearly, **nmove** must be greater than or equal to **nwrcon** - you might want it to be greater if you need a longer run to get the error bar acceptable - this is useful for judging the success of an optimization at each stage.). Also **corper** should be set to a high value (≥ 10) in order that the configurations are completely uncorrelated. The configs generated are written to a file called `config.in`. These configurations are then read and fed into the variance minimizer through a CASINO run with **irun** set to 4. The optimizer then adjusts the value of the parameters in such a way that the variance is minimized.

At this stage, the configurations are distributed according to the original unoptimized wave function. Thus it is often a good idea to recalculate the configurations using the new optimized wave function to generate a new `config.in` and then perform the minimization again. This iterated procedure can typically be carried on as long as desired, but doing more than two or three such iterations is normally of little benefit.

The whole variance minimization procedure is actually automated by the *runvarmin* script, so the two main things to worry about are 'how many configurations to use?' and 'how many parameters to include in the Jastrow factor?'. Good advice about this is given in section 10.23. Basically you should use as many configurations as possible (i.e. greater than 10000 in general) remembering that **nwrcon** refers to the number of configs *per node*. So if you have a ten-node parallel computer, then you can use 10000 configurations by setting **nwrcon** to 1000. Conversely, you should use as few parameters as you can get away with. This might be surprisingly few. Also, don't forget to make sure that the config-generating run has equilibrated before you start to write out configurations (i.e. **nequil** should be suitable high - i.e. at least 1000 and probably more). Use the *plot_vmc_energy* utility and *xmgrace* to check this visually.

The parameters in the Jastrow factor (see section 7.2 are of multiple types (the u term, the χ term and the f term together with their associated cutoffs are the ones normally varied).

If the non-linear cutoff parameters are not being optimized, it is possible to greatly accelerate the optimization process by turning on 'linear mode'. This is done by setting the **vm_mode** keyword to 'linear'. This is expensive in memory however, and if this is a problem, or if the cutoff parameters are to be included in the optimization, then **vm_mode** should be set to 'direct' [NOTE : NOT IMPLEMENTED FOR NEW JASTROW FACTOR - ONLY FOR THE OLD `jasfun.data` FORM (MDT, 6.2004)..]

So let's play. To see a particular example, go and look in `examples/electron_phases/3D_fluid`. This is a homogeneous electron gas calculation with $r_s = 1$ and 54 electrons per cell. Temporarily remove the old Jastrow factor and copy `examples/generic/new_jastrow/jastrow.data_blank_u_chi` to this directory (rename it to `jastrow.data`). The variable parameters are not given explicitly in this example file, so CASINO assumes they are all zero and generates the initial configurations from the Hartree-Fock wave function.

Edit the blank `jasfun.data` file. First change the title from 'Insert title here' to 'Homogeneous electron gas (r_s=1.0)'. Then delete all the lines from 'START CHI TERM' to 'END CHI TERM' (we have just deleted the atom-centred χ parameters, which is sensible because we don't have any atoms in a homogeneous electron gas). For this first example, we don't want to optimize the non-linear cutoff parameter, because it can take a long time to optimize so let's just accept the default. To do this, go to the point after the line where it says 'Cutoff ; Optimizable (0=NO; 1=YES)', set the Optimizable flag to 0. How many parameters do we want to use? Not many, since we're just playing, so where it says 'Expansion order' set the value to 4.

Now edit the `input` file. Let's do 1000 configurations (**nmove** = **nwrcon** = 1000 (Note : this is in general not enough - you should normally have at least 10000 - but we're just playing). Let's use **corper**= 10 to reduce serial correlation. Make these changes now.

Now type *runvarmin -n 2 -v*. The -n flag means 'do two iterations of the config generation/optimization cycle' ; the -v flag means 'when you've finished these two iteration, do a final VMC calculation with the optimized Jastrow factor so we can check that it really does lower the variance' (sometimes it doesn't!). (Note : if you just type *runvarmin* then the default on workstations is equivalent to *runvarmin -n 3 -v*).

The *runvarmin* script will create a directory *io* in which all the input and output files from the different stages of the calculation will be placed (with a numeric suffix indicating which iteration of config generation/optimization they are associated with). The most important things to look at are `vmc.out.1`, `vmc.out.2` and `vmc.out.3`. These are the VMC output files with, respectively, the unoptimized Jastrow factor, the optimized Jastrow factor after the first iteration, and the final optimized Jastrow factor (obviously there will be even more of these if you increase the argument to the -n flag). If you type *ve vmc.out.** etc.. then you can quite clearly see whether the optimization has worked at each stage. Be careful! It won't always work, and you should choose the best of the Jastrow factors from the different iterations (from `jasfun.out.1`, `jasfun.out.2` etc.. in the *io* directory). In this case we see :

```
File vmc.out.1:
Total energy :  0.643382740589 +/- 0.002557624213 Time :  13.8200 seconds
File vmc.out.2:
Total energy :  0.613263097995 +/- 0.001254910540 Time :  21.1500 seconds
File vmc.out.3:
Total energy :  0.611320944695 +/- 0.001157863567 Time :  20.9800 seconds
```

We see that optimizing the parameters in this way lowers the error bar and total energy significantly, and also that most of the improvement is in the first iteration. One could try additional things, such as optimizing the cutoff, or using more parameters, or having different functional forms between different spin types to optimize the Jastrow still further. Feel free to try this.

Attempting to do this more carefully (for me) gave :

```
HF 0.644241947395 +/- 0.000416713194
VMC 0.601693183856 +/- 0.000110549234
DMC 0.599042511676 +/- 0.000185847226
```

where it can be seen that VMC gives around 94% of the correlation energy, if we assume the DMC result is the correct answer.

If we were optimizing a real system with atoms, then we would have included the atom-centred electron-nucleus χ term (one for each type of atom), and possibly the electron-electron-nucleus f term (again, one for each atom type). There are two types of additional term, p and q, but these are rarely used and then only for periodic systems.

6.2.8 How to do a DMC calculation

Diffusion Monte Carlo calculations are the main point of doing QMC. They are generally extremely accurate - comparable or better than the best quantum chemistry correlated wave function techniques - and yet remain applicable to very large systems. However, they require an accurate trial wave function to be efficient. This is normally taken to be a VMC Slater-Jastrow wave function with the parameters in the Jastrow factor optimized with a variance minimization procedure.

So we begin by assuming we have an `input` file, an `xwfn.data` file containing the determinantal part of the wave function, and an *optimized* `jastrow.data` file containing the Jastrow factor. A DMC calculation then consists of three basic steps :

- VMC config generation
- DMC equilibration
- DMC statistics accumulation

The wave function in DMC is not represented in terms of a basis set, but by the time-dependent distribution in configuration space of a set of configurations (or 'walkers'). The shape of the many-electron wave function in configuration space is built up by killing or duplicating individual walkers according to certain rules, and thus the population of walkers fluctuates.

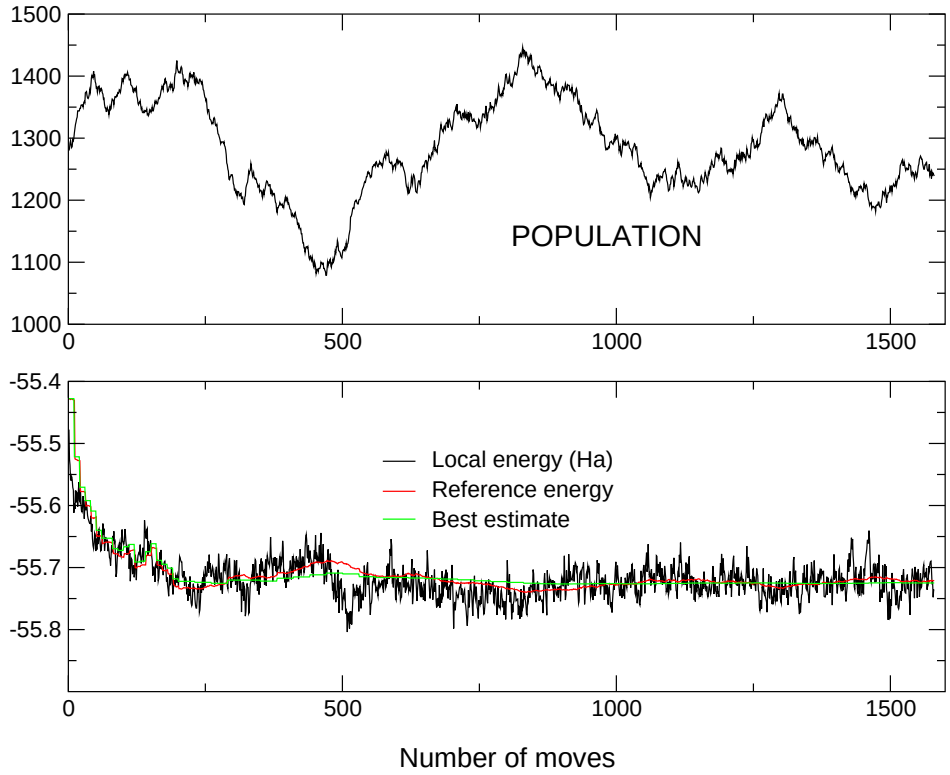
The first step in DMC then is to generate these configurations in their initial distribution (i.e. according to the VMC trial wave function). This is done through a VMC calculation in exactly the same way as we generated configurations for variance minimization in the previous section. Therefore one should again set **nwrcon** to be the desired number of configs/node, and **nmove** to be $\geq$ **nwrcon**. (Setting the correlation period **corper** to a higher value than in VMC is less important than in variance minimization, as the DMC configs are time dependent). Note - this is a common point of confusion - that the **irun** flag should be set to 2 (VMC with Jastrow) at the start of a DMC calculation not, as might be thought, to 3 (DMC). When it is required to actually carry out the DMC part of the calculation after the configs have been generated, the *rundmc* script will automatically flick **irun** from 2 to 3.

An appropriate number of configs to use for DMC might be 640 (10 configs per node on a 64-node parallel machine) or more. You can usually get away with using 200 configs on a single-processor workstation for small systems - this will clearly start getting too expensive for a single workstation quite quickly as you increase the system size. Parallel machines are in general a good thing.

Following config generation, the distribution of walkers is then allowed to propagate in imaginary time according to the usual DMC rules. During a certain period of equilibration, the distribution will change until the walkers are distributed according to the ground-state wave function of the system, subject to the constraint that the wave function has the same nodes as the VMC trial function that we started with. This part of the process is called 'DMC equilibration'. The best estimate of the energy will fall from the initial VMC value to around the correct ground state energy during this process. After equilibration, the best estimate of the energy will be roughly correct, and we must now propagate for a long period of imaginary time to accumulate enough energy data to estimate the DMC energy with a low enough error bar. This is the statistics accumulation part.

During its operation, the *rundmc* script creates a directory called `io` containing all the `input`, `out` and `config.in` files from the various stages of the calculation renamed according to an obvious scheme (e.g. `out_congen`. `out_equil` and `out_stats`). The important files (left in the original working directory) are `dmc.hist` and `dmc.hist2` which contain energy and other data as a function of move number (see sections 7.10 for precise definitions of what they contain - note that `dmc.hist` contains the most important data and `dmc.hist2` the less important).

The progress of a DMC simulation is easily visualized by means of the utility *graphit*, which reads the `dmc.hist` file and calls the plotting programs *xmgr* or *xmgrace* to show you the resulting pretty picture. This requires you to have *xmgr* or *xmgrace* set up on your machine - see `plasma-gate.weizmann.ac.il/Grace/` if you haven't. Typing *graphit* will produce something like this :



This picture shows the results of the simulation of an antiferromagnetic NiO crystal with 10 configs per node on 128 nodes of a parallel computer. The upper panel shows how the population fluctuates as the simulation progresses. There is a feedback process in operation to limit the population fluctuations so this should just oscillate around the total initial number of configurations that we chose (1280 in this case). If you were smart and printed this manual out in colour, then in the bottom panel, you will see a red line, a green line, and a wobbly black line. The black line is the instantaneous value of the local energy (the thing which gets averaged over), the red line is the reference energy E_T which is adjusted to control the feedback process that keeps the population in check; the green line is the best estimate of the DMC energy as the simulation progresses.

You should note that in the picture above, the best estimate of the energy falls from its initial value from the VMC config generation run to a much lower constant value (around -55.72 a.u.) as the wave function evolves to the ground state. You should look at what point the energy becomes roughly constant i.e. at around 500 moves in this case. This splits the graph quite neatly into an equilibration phase and a statistics accumulation phase. When we average energies to produce the final energy and error bar, we should only include those moves between 500 and the end of the run.

To see DMC in action, go to examples/molecules/h2/RHF/dmc and let's calculate the DMC energy of a hydrogen molecule. The input files should already be set up correctly. We see that an equilibrated VMC run is set to go for **nmove**= 250 moves in order to produce **nwrcon**= 250 configs. For DMC equilibration, we run 100 blocks (**nblock_dmc_equil**) of 10 moves (**nmove_dmc_equil**). For the statistics accumulation we run 200 blocks (**nblock_dmc_stats**) of 200 moves (**nmove_dmc_stats**). The DMC **nconfig** parameter is set to 250 - the same as **nwrcon**. The only reason **nconfig** and **nwrcon** would ever differ is on a parallel machine; config generation is very fast, and can be done quickly on far fewer nodes than the DMC calculation. Running config generation on batch queues with small numbers of nodes thus usually reduces time spent waiting in queues; increase **nwrcon** in proportion to the reduction in the number of nodes to maintains the same total config number. The **dtdmc** parameter is the DMC timestep - note that it is much smaller than the VMC timestep (it needs to be small because the DMC Green's function is only exact in the limit as the timestep goes to zero). A typical value in a DMC simulation might be 0.003. Note that for accurate work, you need to consider extrapolating to zero timestep (see the utility *extrapolate_tau*). Note finally that **irun** is set to 2 - indicating that we do a VMC calculation first. Now type *rundmc* and wait till the file

DMC_FINISHED appears.

Amuse yourself by looking at the output files in the *io* directory. Now type *reblock* in the directory where the dmc.hist and dmc.hist2 files are (reblock also reads input to determine some input parameters it needs to know about - it will ask you about these if it can't find input). If all files are present, *reblock* will read these files and then starting asking you questions. What units do you want the answer in? Whatever you want. At which line of dmc.hist do you want to start computing statistics? The move number where the green line in the graphit output becomes approximately constant (e.g. 501 in the NiO case). Average over how many lines? Usually input '0', meaning use the rest of the (now equilibrated) data in the file from lines 501 to the end. Reblock will then show you a reblocking analysis, and ask you to choose a block length. Choose a value where the 'Std err of mean' column starts to plateau (again - the *plot_reblock* utility can help you with this). A value of 64 might be typical. Reblock will then print out the final DMC energies and error bars, together with an analysis of the population fluctuations, effective time steps, and acceptance ratios.

In the case of molecular hydrogen, the final energy is -1.1744731 ± 0.00056. The exact energy is -1.1744757 a.u. so this result is pretty good without paying particular attention to getting it absolutely right, which is promising.

6.2.9 How to generate localized orbitals

For large systems a significant speed up may be gained by using *localized orbitals* in CASINO instead of the canonical delocalized orbitals generated by the DFT/HF code. Currently this facility is not fully implemented and such orbitals may only be generated by post-processing a pwfn.data file (a localized orbital facility for Gaussian basis sets will be added soon). Additional limitations mean that currently only cuboidal simulation cells with a single k point ($\mathbf{k} = \mathbf{0}$) for spin-unpolarized systems may be treated. A detailed discussion on the use of localized orbitals in QMC will appear in the Theoretical Background section fairly soon.

Let's now look at the procedure for setting up a calculation with localized orbitals using the silane molecule (SiH_4) as an example. The required input files have been set up appropriately in CASINO/examples/molecule/splines/silane_localized.

It is assumed that the user is trying out this example on a serial workstation. (The generate_spline utility can be run in parallel: an appropriate job submission script has to be used.)

1. **Carrying out the transformation from Bloch to Wannier orbitals**. The first stage requires the pwfn.data and input_wannier[1] files. Type *xwannier* to run the wannier conversion utility. The output consists of a new plane-wave file pwfn.data.wannier and a file wannier_centers.dat that contains a list of the Wannier function centres.

2. Type *mv pwfn.data pwfn.data.bloch* and then type *mv pwfn.data.wannier pwfn.data*. From now on, we will work with localized orbitals.

3. **Generation of a spline representation of the orbitals**. We could run CASINO using the localized orbitals represented in the plane wave basis; however, it is faster and more memory efficient to represent the localized orbitals numerically. The next stage requires a pwfn.data file (and corresponding wannier_centers.dat file if this exists) and a generate.dat[2] file. Type *generate_spline* to run the utility for producing the splined orbitals. The output of this code is a swfn.data file[3].

[1] The only parameter the user is ever likely to want to change in input_wannier is the number of states included in the Wannier transformation (all the states in the pwfn.data file from 1 up to this number are transformed; any remaining states are unchanged). In the present case, we are including all the states in the transformation.

[2] For this example, we shall use localized grids for all the splined states; if, however, some states were non-localized then it would be advisable to represent those states over the whole simulation box. The "multiplication factor" in the generate.dat file controls the fineness of the spline grid. Generally, this needs to be set to at least about 2 before the kinetic energy of the splined orbitals is the same as that of the orbitals in the plane-wave basis to an high degree of accuracy. The orbitals have a spherical truncation surface such that 99.7% of the square of the orbital norm lies inside this radius. A further shell of width at least 0.05 a.u. is saved from truncation: the orbital can be brought smoothly to zero over this shell region.

[3] The swfn.data file is unformatted. If you wish to look at its contents then use the format_spline utility to generate a formatted swfn.data.formatted file. This is also useful if you want to transfer a swfn.data file from one platform to

4. **Carrying out a QMC calculation.** Now, finally, we can carry out QMC calculations using the localized orbitals. For example, type *runvmc* to carry out a HF VMC calculation. The usual CASINO inputs[4] are required, along with the `swfn.data` file.

6.3 How to run the code : run scripts

As you may have already gathered, although CASINO can be run by sitting in a directory containing input files and typing *casino* > & *out*, it is recommended that one run it using the shell scripts provided in the CASINO/scripts directory.

There are three basic scripts, meant to control VMC, DMC and variance minimization calculations. Predictably their names are *runvmc*, *rundmc* and *runvarmin*. These may be different for different machine types, but set-up should be automatic following definition of the QMC_ARCH environment variable i.e. the relevant scripts for your machine type should be copied into your path on typing *make* in CASINO/utils.

The scripts are designed to reduce the effort of doing QMC calculations to just one command. They are most useful when using parallel machines with a batch queueing system. They are written to detect all common errors that a user may make in setting up a calculation, so that the fault is detected immediately rather than when the batch job starts (which may be many hours later). If the job would take longer than the maximum 'time slot' on such a machine, the scripts allow you to 'chain' jobs, so that for example typing *runvmc -n 5 16* will run 5 successive VMC jobs on a 16 node queue with (say) an 8-hour time limit on each job, with the script handling all the required i/o decisions. The result is 40 hours worth of statistics.

DMC calculations and variance minimization calculations are more complicated, since they require you to submit different types of job within a single calculation (DMC: config generation/equilibration/statistics accumulation), VARMIN: n iterations of config generation/variance minimization). These are handled automatically with all changes to the input files being performed by the scripts. For example, the first step in a DMC calculation is to perform a VMC calculation to generate a bunch of configurations distributed according to the square modulus of the trial wave function. The next step is to use DMC to equilibrate the configurations. Amongst other things, the rundmc script will automatically change the 'IRUN' flag in the input file from 2 to 3, telling CASINO to switch to a DMC calculation.

See the usage notes at the top of each script for precise information about how they work, or, on parallel batch machines at least, just type the name of the script to get a usage note. With all scripts, there are normally -debug, -prof and -dev flags to run the versions of the code compiled with different compiler options which are automatically stored by the Makefile in different bin directories.

Note that, on a batch machine, the script will create a batch queue file for the first job in the list, show it to you, and ask you whether it should submit it. You should answer 'y' or 'n'. Then, if you have requested that the script perform some action in the future (when the job it has just submitted has finished), then the script will stay alive after submitting the job and demand to be put in the background. Do this by holding down the control key and pressing 'z' to suspend the script, followed by the command *bg* to put it in the background. The script will lurk and monitor the queues every so often to check when the current job has finished, then perform the appropriate actions to submit the next job in the sequence and so on.

The following notes try to demonstrate typical usage of these scripts on various types of machine:

---

another, since the unformatted file can usually only be read by the machine that generated it. Run `format_spline` in the presence of `swfn.data.formatted` file to produce the corresponding unformatted file.

[4] Note the following in the `input` file: 1. We have set `isperiodic` to false because we are simulating a molecule. Now that we have generated truncated localized orbitals, we can dispense with periodicity; 2. We have provided lists of the occupied states for spin-up and spin-down electrons (the `statelist_up` and `statelist_down` blocks).

Workstations/parallel machines without batch queues:

```
runvmc
runvmc -big
runvmc -nnodes 4
```

Do a VMC calculation. The -big option will create the potentially large vmc.hist files in a pre-defined scratch space and create a link to them. There is not usually a -n option to chain multiple jobs, because there is no time limit on a typical workstation. The -nnodes flag says how many nodes to use on parallel machines.

```
runvarmin -n 2 -v
runvarmin -n 2 -v -nnodes 16
```

Run 2 iterations of a config generation/variance minimization cycle (4 CASINO jobs altogether). The -v flag tells it to run a post-optimization VMC calculation with the final optimized Jastrow. The optional -nnodes flag says how many nodes to use on parallel machines.

```
rundmc
rundmc -c -e -s
rundmc -s
rundmc -nnodes 32
```

Run a DMC calculation. Default is to do one config generation, one DMC equilibration, one DMC stats accumulation, though this may be changed by supplying the -c -e or -s flags explicitly. Let's say you have completed a DMC calculation, but the standard error is too large. You can accumulate more statistics on top of the ones you already have by typing 'rundmc -s'. Remember to do a DMC calculation with the rundmc script, you need to setup your input file to do a *VMC* config generation. All parameter changes to run DMC equilibration and stats accumulation will then be performed automatically by the script. The optional -nnodes flag says how many nodes to use on parallel machines.

Parallel machines with batch queues:

Depends entirely on the computer architecture. In all cases, just typing the name of the script will present you with a usage note, detailing exactly what options, queues, nodes and time limits are available.

```
runvmc -n 2 16
runvmc -n 2 16 night
```

Chain 2 successive VMC jobs on 16 nodes. Some machines require an extra parameter such as test/short/medium/long or day/night/weekend or some such, to indicate which 16 node queue you want to submit to. 'test' might denote a 10 minute queue, 'weekend' a 24 hour one etc..

```
rundmc -c -e -s 4 16
rundmc -c 4 test -e -s 4 16 long
```

Run 1 config generation, 1 DMC equilibration, followed by 4 chained DMC statistics accumulation jobs on 16 nodes. Again, some machines might require test/short/medium/long. Note that some scripts will allow you to run DMC config generation jobs—which are generally very quick—on a small number of nodes (say 4) with a short time limit, then do the actual DMC on lots of nodes with a long time limit which is what the second example above does. Note that it is not currently possible to use the scripts to chain multiple DMC equilibration jobs—only 1 is allowed. This behaviour could be simulated through multiple 'rundmc -e' jobs, but the scripts will be changed to do this properly eventually—it's only important for incredibly large systems anyway).

```
runvarmin -n 3 16
runvarmin -n 3 16 night
```

Run 3 iterations of a config generation/variance minimization cycle on 16 nodes. Again, some machines might require you to specify test/short/medium/long etc..

Note if you make any changes to the scripts in the appropriate CASINO/scripts/$QMC_ARCH directory, typing *make* in the utils directory will copy the modified scripts to the appropriate bin_qmc directory which should be in your path.

There is also a *runvarmin1* script on some machines, which is designed for machines where you can waste weeks sitting around in batch queues (such as Cambridge HPCF Origin 2000 machines). Basically what it does is to generate complicated batch scripts which will do lots of separate CASINO jobs within a single allocated time slot in some queue. See the top of the script for more information.

Note that ultimately, the scripts for the different machine types wil be merged into two—one for batch queuing systems, one for interactive use, but as this is deeply tedious I keep putting it off.

7 Files used by CASINO

CASINO uses a variety of files during the course of its operation to store things in. Here is a list of them:

Input Files

input	Parameters controlling the QMC calculation (See Section 7.1)
xx_pp.data	Pseudopotential, if required (xx=element symbol in lower case e.g. si, c) (See Section 7.3)
jastrow.data	Jastrow factor, if required (See Section 7.2)
xwfn.data	(x=g,p,b,a) Trial wave function in Gaussians, plane waves, blips or on a grid.
heg.data	Control file for electron and electron-hole systems

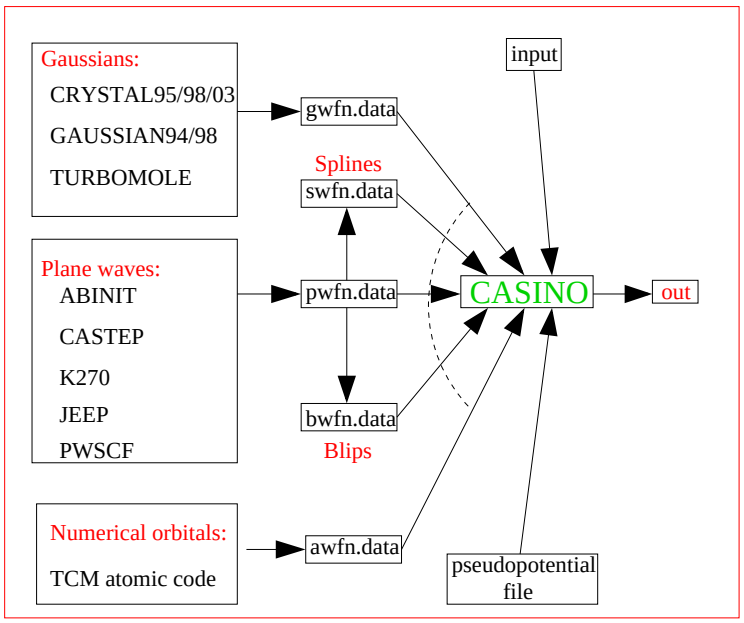
Mandatory : input plus either an xwfn.data file or a heg.data file. Others optional.

In/Output Files

eepot.data	Fourier transform of $1/r$—required for the MPC interaction Generated by CASINO with IRUN=5 and used in subsequent runs.
density.data	Charge density in Fourier space—required for the MPC interaction. Generated by CASINO with IRUN=6 and used in subsequent runs.
vmc.posin/vmc.posout	List of final electron coordinates on all nodes
config.in/config.out	List of positions/energies etc. of a set of configurations – required by DMC and varmin

Output Files

out	CASINO output file (sent to standard out : named 'out' by the run scripts)
vmc.hist	VMC history file (See Section 7.9)
dmc.hist	DMC history file, most important info (See Section 7.10)
dmc.hist2	DMC history file, less important info (See Section 7.10)



Input Files:

The contents and format of the various input and output files will now be described.

7.1 Basic input file : input

The file 'input' contains all the parameters needed to control the QMC calculation. A complete list of the input parameters is given below. Further details, including default values, may be found by using the *casinohelp* utility. Type 'casinohelp all' to get a list of all keywords that CASINO knows about, or 'casinohelp keyword' for detailed help on a particular keyword. Type 'casinohelp search *text*' to search for the string *text* in all the keyword descriptions. Create a blank file containing the keyword 'input_example' and type 'runvmc'—this will create a sample input file containing all valid input keywords and their default values in the correct format.

The input file is meant to be very flexible and the list of understood keywords can vary with time without breaking anything. See the comments at the top of the 'esdf.f90' module for more information.

- Each line is of the form 'keyword : value'. There *must* be a space either side of the colon.
- The parameters are divided into types : string/integer/single/double/physical/boolean. Variables of 'physical' type must be supplied with a unit, such as 'hartree', 'eV', 'rydberg' or Joules/Megaparsec. All reasonable physical units are understood.
- The parameter names are case insensitive (e.g. EnerGyCuToFF is equivalent to energycutoff) and punctuation insensitive (energycutoff is equivalent to energy_cutoff is equivalent to energy-cutoff). Punctuation characters are '.', '_', and '-'.
- Some of the parameters are of 'block' type which means that multiple parameters must be supplied which may be spread over several lines. See, e.g., 'wavefunction' or 'lineplot'.

Here is the current list of the input parameters in alphabetical order. Note that the *casinohelp* facility is always up to date (it runs CASINO to find out what it knows about) - but this manual may not be.

Input parameters:

ALIMIT (*Real*) Parameter required when LIMDMC=2 or 3. A value of 0.25 was suggested by Umrigar et al. for all-electron calculations, but a value of 1.0 may be more appropriate for pseudopotential calculations. The answers are insensitive to the precise value. ALIMIT is not required if NUCLEUS_GF_MODS is set to true. See Section 10.6.

BTYPE (*Integer*) Determines type of basis set. Possible values:
1: Plane-waves (input from CASTEP, PWSCF, ABINIT, GP, k207);
2: Gaussians (from CRYSTAL95/98/03, GAUSSIAN94/98/03, TURBOMOLE);
3: Numerical (atoms only);
4: Blip function (B-spline) basis (from transformation of pwfn.data using the *blip* converter).;
5: Gaussian Wannier orbitals (non-functional).
6: Spline basis (from transformation of pwfn.data using the *generate_spline* converter)
Note that the definition of BTYPE was changed in December 2003. This affected only blip and spline inputs, and (obviously) the various electron-hole phases.

CALC_VARIANCE (*Logical*) If CALC_VARIANCE is true then the variance of the local energy is calculated in each block of VMC. If several block are used then the block variances will be averaged and the standard error in the variance will be calculated. The variance calculation assumes that the individual energies are uncorrelated, so CORPER should be given a large value. Note that VMC_METHOD must be 1 if the variance is to be calculated.

CEREFDMC (*Real*) Constant used in updating the reference energy in DMC algorithm. See Section 10.5.

CHECKWFN (*Logical*) Enable a numerical check of the analytic orbital derivatives coded in the various routines such as gaussians_periodic/gaussians_mol/bwfdet/pwfdet etc.

CONFIGS_VERBOSE (*Logical*) Setting CONFIGS_VERBOSE to .true. will cause the printout of condata_xxx files on each node, where xxx is the id number of the node. These files contains extra information about the energies of generated configs which is useful in debugging and so on but is not normally required.

CON_LOC (*Character*) Directory to which configurations written.

CORPER (*Integer*) VMC only. Local energy calculated only every CORPER moves.

CUSP_CORRECTION (*Logical*) When expanded in a basis set of Gaussian functions, the electron-nuclear cusp present in all-electron calculations is not represented correctly, since the gradient of a Gaussian is necessarily zero there. Clearly this only matters for s-type GTFs, since all functions of higher angular momentum have a node at the nucleus. When the CUSP_CORRECTION flag is activated, each orbital expressed as a sum of s-type Gaussians is replaced within $r = r_c$ by ISIGN$*exp\,(p(r))$, where $p(r)$ is the polynomial $p = \alpha_1 + \alpha_2 r + \alpha_3 r^2 + \alpha_4 r^3 + \alpha_5 r^4$ such that $p(r_c)$, $p'(r_c)$, $p''(r_c)$ are continuous, $p'(0) = -Z_{nuc}$. $E_L(0)$ is set by a smoothness criterion. This procedure can be expected to greatly reduce fluctuations in the local energy in all-electron Gaussian calculations. See Section 10.13 for more details.

CUSP_INFO (*Logical*) If CUSP_CORRECTION is set to true for an all-electron Gaussian basis set calculation, then CASINO will alter the orbitals inside a small radius around each nucleus in such a way that they obey the electron-nuclear cusp condition. IF CUSP_INFO is set to true, then information about precisely how this is being done will be printed to the output file. Be aware that in large systems, this may produce a lot of output. Furthermore, if you create a file called 'orbitals.in' containing an integer triplet specifying which orbital/ion/spin you want, the code will additionally print graphs of the specified orbital, radial gradient, Laplacian and 'one-electron local energy' to the files orbitals.dat, gradients.dat, laplacians.dat and local_energy.dat. These graphs may be viewed using xmgr/grace or similar plotting programs. See Section 10.13 for more details.

DBARRC (*Integer*) Maximum number of blocks between recalculation of DBAR matrices (see Section 10.15).

DENFT_THRESHOLD (*Real*) When calculating the Fourier transform of the charge density and writing the result to the density.data file (which CASINO will do if you set IRUN=6) the program only writes data for those **G** vectors where the absolute value of either the real part or the imaginary part of the Fourier coefficient is larger than a threshold. This is that threshold. The default of 10^{-6} should be good enough.

DTVMC (*Real*) Time step for VMC run (atomic units).

DTVMC2 (*Real*) Time step for VMC run (atomic units). In an electron-hole calculation, DTVMC is used to move the electrons while DTVMC2 is used to move the holes.

DTDMC (*Real*) Time step for DMC run (atomic units).

EDIST_BY_ION, EDIST_BY_IONTYPE (*Block*) The optional EDIST_BY_ION block allows fine control of the initial distribution of the electrons before equilibration starts. The standard algorithm shares out the electrons amongst the various ions weighted by the pseudo-charge/atomic number of the ion. Each electron is placed randomly on the surface of a sphere surrounding its parent ion. There are certain situations, for example a simple crystal with a very large lattice constant, where the standard algorithm in the POINTS routine may give a bad initial distribution which cannot be undone by equilibrating for a reasonable amount of time. This keyword allows a user-defined set of electron/ion associations to be supplied. The syntax is to supply N_ION lines within the block which look like, e.g., 1 4 4, where the three numbers are: the ion sequence number; the number of up-spin electrons associated with this ion; the number of down-spin electrons associated with this ion. Alternatively one may use the EDIST_BY_IONTYPE keyword block where you replace the ion sequence number with the ion type sequence number and the information is supplied only for each particular type of ion.

EH_SAFEMODE (*Logical*) In some cases the electron-hole distribution resulting from a variance minimization run can yield completely wrong VMC results. It is found that the problem is introduced by a Jastrow factor which pulls electrons and holes together so strongly that the wave function collapses. This problem can be solved by setting EH_SAFEMODE to T, which prevents the e-h component of the Jastrow factor from cancelling the other terms. It is recommended to set VM_SMOOTH_LIMITS=T, OPT_MAXITER≈3, and running many (around 50) variance minimization iterations when using this feature - the *runvarmin* script will allow this if EH_SAFEMODE=T. After this, it is a good idea to set EH_SAFEMODE=F and run a normal variance minimization from the output of the previous one.

ENERGY_CUTOFF (*Physical*) This is the energy cutoff for G vectors used in (a) the FFT of the MPC interaction required when generating eepot.data (IRUN=5), and (b) the FFT of the one-particle density required when generating density.data (IRUN=6). The program will suggest a value for ENERGY_CUTOFF if the existing value is unsuitable, or if the user inputs a value of zero.

E_OFFSET (*Physical*) This keyword gives a constant shift in the total energy such that the the final result will be $E = E_{calc} - E_{offset}$. The default is zero.

ESUPERCELL (*Logical*) By default total energies and their components are printed as energies per primitive cell. Switching this flag to T forces printing of energies per simulation cell in the output file.

ETYPE (*Integer*) Determines type of electron-hole phase (assuming btype==0). Possible values:
1: Homogeneous electron gas (fluid phase: plane wave basis);
2: Homogeneous electron gas (crystal phase: Padé function basis);
3: Homogeneous electron-hole gas (fluid phase: plane wave basis);
4: Homogeneous electron gas (excitonic insulator phase: exponential pairing basis);
5: Homogeneous electron-hole gas (crystal phase: Padé function 6: Quasi-2d jellium slab (fluid phase with hard walls).

EWALD_CHECK (*Logical*) CASINO and the wave function generating program should be able to calculate the same value for the nuclear repulsion energy, given the same crystal structure. By default CASINO therefore computes the Ewald interaction and compares it with the value given in the wave function file. If they differ by more than 10^{-5}, then CASINO will stop and complain. If you have a justifiable reason for doing so, you may turn off this check by setting EWALD_CHECK to F.

EWALD_CONTROL (*Real*) This is the percentage increase (from the default) of the cutoff radius for the reciprocal space sum in the Ewald interaction - used for calculating electrostatic interactions between particles in periodic systems. Its default value is zero. Increasing this will cause more vectors to be included in the sum, the effect of which is to increase the range of the Ewald gamma parameter over which the energy is constant (the default gamma should lie somewhere in the middle of this range). This need only be done in exceptional circumstances and the default should be fine for the general user.

FIXED_E_POS (*Block*) This is the position r at which to fix an electron during calculation of the pair correlation function g(r,r') activated with the PAIR_CORR keyword. To be given in atomic units.

GAUTOL (*Real*) Tolerance for Gaussian orbital evaluation. Neglect contribution if $\exp(-\alpha r^2) < 10^{-\mathrm{GAUTOL}}$.

GROWTH_ESTIMATOR (*Logical*) Turn on calculation of the growth estimator of the total energy in DMC calculations. The difference between mixed estimator and the growth estimator for the energy provides a rough estimate of the time-step bias.

IACCUMULATE (*Logical*) .TRUE. for accumulation of averages. Normally turned off during equilibration.

IBRAN (*Logical*) If set to .TRUE. then enables weighting and branching in DMC. IBRAN=.FALSE. may be used to check the DMC algorithm.

IDEN (*Integer*) Controls writing of Fourier-space charge density to file:
1: Density appended to file density.data (or written to mol_density.data in finite systems);
0: Density not written out.

INEW (*Integer*) Determines whether this is a new run or a continuation of a previous one:
1: (VMC) Start a new run;
(DMC) Read old VMC configurations and recalculate the current best estimate of the ground-state energy as the mean of the configuration energies;
0: (VMC) Continuation of old run. Read in old configurations;
(DMC) Continuation of old run. Do not recompute the current best estimate of the energy.

INPUT_EXAMPLE (*Logical*) If INPUT_EXAMPLE is .TRUE. then write out an example of a QMC input file with all currently known keywords and their default values. A modified version of this can be used as an input file in future runs.

IRUN (*Integer*) Determines type of calculation. Possible values:
1: VMC run with wave function of determinant(s) only;
2: VMC run with a Slater-Jastrow wave function;
3: DMC run;
4: Variance minimization run (for optimizing a trial wave function);
5: Generate eepot.data file for geometry in xwfn.data (x=p, g), for MPC interaction. See Section 10.20.4;
6: Generate density.data file for geometry given in xwfn.data (x=p, g).

ISOTOPE_MASS (*Real*) Nuclear mass override for the relativistic corrections activated by RELATIVISTIC. This keyword can be used to define the nuclear mass (in amu) if you need to override the default value (which is averaged over isotopes according to their abundances). The default (given in the table in Section 10.27) is used if ISOTOPE_MASS is set to zero. The atomic mass unit (amu) in this sense means 'the ratio of the average mass per atom of the element to 1/12 of the mass of 12C'.

ISPERIODIC (*Logical*) .TRUE. if and only if system is periodic in 1, 2 or 3 dimensions.

ITERAC (*Integer*) Determines type of electron-electron interaction:
1: Ewald interaction (see Section 10.20) or $1/r$ for finite systems;
2: MPC interaction (see Section 10.20.4);
3: Ewald and MPC. Ewald used in DMC propagation;
4: Ewald and MPC. MPC used in DMC propagation.

JASBUF (*Logical*) If JASBUF is .TRUE. then the chi term in the Jastrow function for each electron in each configuration is buffered: saves time at the expense of memory. Clearly this will have no effect in systems without one-body terms in the Jastrow, such as the HEG.

JASTROW_PLOT (*Block*) This utility allows you to plot $u(r_{ij})$, $\chi(r_i)$ and $f(r_i, r_j, r_{ij})$ terms in the new Jastrow factor. The first line is a flag for whether the Jastrow factor is to be plotted (0=NO, 1=YES); the second line holds the spin of particle i; the third line holds the spin of particle j; the fourth line holds the (x, y, z)-position of particle j (for plots of f); the fifth line holds a vector with the direction in which i is moved (for plots of f); and the sixth line holds the position vector of a point on the straight line along which electron i moves (for plots of f). Note that the nucleus is assumed to lie at the origin. If f is plotted, the `jastrow_value_f_?.out` files contain the value of f against the distance from the point given in line 6.

KWARN (*Logical*) The KWARN flag is relevant only in calculations using a plane wave basis set. If the flag is set to .TRUE., then the routine PWFDET_SETUP will issue a warning whenever the kinetic energy calculated from the supplied orbitals differs from the DFT kinetic energy given in the pwfn.data file by more than an internal tolerance (usually set to 10^{-6}). If the flag is .FALSE., then CASINO will stop with an error message on detecting this condition. Note

that in cases where the DFT calculation which generated the orbitals used fractional occupation numbers, the kinetic energy mismatch is very likely to occur since QMC deals in principle only with integer occupation numbers, hence the existence of this flag.

LCUTOFFTOL (*Real*) This is used to define the cutoff radius for the local part of the pseudopotential. It is the maximum deviation of the local potential from $-z/r$ at the local cutoff radius.

LIMDMC (*Integer*) Set modifications to Green's function (see Section 10.6.) May take values:
0: no modifications applied;
1: Depasquale *et al.* limits to drift-vector and energy;
2: Umrigar *et al.* modifications (need to supply ALIMIT parameter);

LINEPLOT (*Block*) This utility allows you to plot the value of certain quantities (such as orbitals and their derivatives, or various local energy terms) along a line from point A to point B. The data will be plotted in a format suitable for xmgr/grace in the file `lineplot.dat`. The syntax of the block is as follows : LINE 1: what_to_plot. This can be : 1=orbitals, 2=grad_x of orb, 3=grad_y of orb, 4=grad_z of orb, 5=laplacian of orb, 6=local e-i potential, 7=local energy with $N-1$ electrons fixed in random positions (or specified positions - see later).
Then for what_to_plot=1-5 add lines 2-7 ; for what_to_plot=6, add lines 5-7 ; for what_to_plot=7, add lines 2a,3a+,5-7.
LINE 2: number of orbitals *norb*
LINE 3: *norb* orbital sequence numbers identifying the orbitals to be plotted
LINE 4: spins (1 or -1) for each of the orbitals in the previous line
LINE 5: the number of points to be plotted along the line
LINE 6: xyz coordinates of point A (in a.u.)
LINE 7: xyz coordinates of point B (in a.u.).
When plotting the local energy, the positions of up to $N-1$ electrons may be fixed (e.g. to investigate e-e coalescences). To do this add the following lines in place of lines 2-4 :
LINE 2a: number of electrons to fix (0 if you don't want to fix any)
LINE 3a+: position of each electron you want to fix (in a.u., one per line).

LWDMC (*Logical*) Weighted DMC only: allow configuration weight to vary between WDMCMIN and WDMCMAX before killing or branching.

MAKEMOVIE (*Logical*) Plot the particle positions every MOVIEPLOT moves (see Section 11).

MAX_REC_ATTEMPTS (*Integer*) This is the maximum number of times DMC will attempt to restart a block if it continues to encounter catastrophes. Relevant only if the TRIP_POPN keyword is set to a non-zero value. See the discussion in Section 10.10 for more details.

MOVIECELLS (*Logical*) If .FALSE. then MAKEMOVIE will plot the unit cell; if .TRUE. then nearest-neighbour cells in the (x, y)-plane will also be written (see Section 11).

MOVIENODE (*Integer*) Plot the particle positions on node MOVIENODE (see Section 11).

MOVIEPLOT (*Integer*) Plot the particle positions every MOVIEPLOT moves (see Section 11).

NBLOCK (*Integer*) The total number of blocks of NMOVE moves in the run. N.B., you should use the REBLOCK utility in the utils directory to see the effect of varying the block size on the variance after the calculation has completed. In VMC NBLOCK just determines how often block averaged quantities are written to the output file. In DMC the configuration population is redistributed among the nodes and may, in extremis, be renormalized at the end of each block.

NCONFIG (*Integer*) DMC only. Initial number of configurations per node. This can be different from NWRCON (number of configs written out by preliminary VMC run) due to the possibility of the config generation and DMC simulation being run on different numbers of nodes on a parallel machine (this might be desirable as the VMC part is often very quick and can be run on relatively few nodes). It is possible to carry out a DMC run with a target weight of NCONFIG=1 configurations per node, provided you are running on more than one node. This is liable to lead to parallel inefficiency, however, especially on small numbers of nodes.

NEIGHPRINT (*Integer*) NEIGHPRINT = n will generate a printout of the first n stars of neighbours of each atom in the primitive cell, with the relevant interatomic distances given in both Angstrom and a.u. If $n = 0$ or if you are an atom-free electron or electron-hole fluid phase, then the keyword has no effect.

NEQUIL (*Integer*) Number of Metropolis equilibration steps. Note that the correlation period is not accounted for, i.e., NEQUIL configuration move attempts are made.

NEWOPT_METHOD (*Text*) NEWOPT_METHOD specifies the method used to minimize the quartic LSF. NEWOPT_METHOD should be one of: 'CG' (conjugate gradients), 'MC' (Monte Carlo), 'LM' (line minimization), 'SD' (steepest descents), 'BFGS' (Broyden-Fletcher-Goldfarb-Shanno), 'BFGS_MC' (BFGS and Monte Carlo), 'CG_MC' (conjugate gradients and Monte Carlo) or 'GN' (Gauss-Newton). (See section 10.24).

NEU, NED (*Integers*) Number of up-spin and down-spin electrons.

NHU, NHD (*Integers*) Number of up-spin and down-spin holes.

NEWOPT_ITERATIONS (*Integer*) NEWOPT_ITERATIONS specifies the maximum number of conjugate-gradients, steepest-descent or BFGS iterations to be performed if NEWOPT_METHOD is 'CG', 'SD', 'BFGS', 'CG_MC' or 'BFGS_MC'. If NEWOPT_METHOD is 'MC', 'LM', 'CG_MC' or 'BFGS_MC' then it specifies the number of line minimizations to be performed. (See section 10.24).

NLCUTOFFTOL (*Real*) This is used to define the cutoff radius for the non-local parts of the pseudopotential. It is defined as the maximum deviation of the non-local potentials from the local potential at the non-local cutoff radius.

NLRULE1 (*Integer*) Rule for non-local integration in production VMC or DMC run, see Section 10.18. Currently assumed to be the same for all atoms if it is controlled through this keyword ; you can provide an override value of NLRULE1 for particular atoms at the top of the corresponding pseudopotential file.

NLRULE2 (*Integer*) Rule for non-local integration in configuration generation for DMC or wave function optimization, see Section 10.18. Currently assumed the same for all atoms if it is controlled through this keyword ; you can provide an override value of NLRULE2 for particular atoms at the top of the corresponding pseudopotential file. It is usual to set NLRULE2 greater than NLRULE1 as the accuracy of the integration is more critical in config generation (errors in the non-local energy bias wave function optimization and DMC calculations), but note that setting it to something different than NLRULE1 can lead to a significant performance disadvantage when VMC_METHOD=1 is used as all the energies have to be computed again.

NMOVE (*Integer*) For DMC NMOVE is the number of moves of all electrons in a block. For VMC the total number of moves of all electrons in a block is NMOVE×CORPER×NVMCAVE.

NPCELL (*Block*) Vector of length three giving the number of primitive cells in each dimension that make up the simulation cell. N.B., for the 1D periodic case, NCELL(2) and NCELL(3) must be 1; for 2D slab case NCELL(3) must be 1.

NUCLEUS_GF_MODS (*Logical*) This keyword is the switch for enabling the use of the modifications to the DMC Green's function for the presence of bare nuclei, suggested in J. Chem Phys. 99, 2865 (1993), expected to reduce timestep errors in all-electron calculations.

NVMCAVE (*Integer*) VMC only. Instead of writing out local energies etc. every time they are calculated (i.e., every CORPERth configuration move), we average over the NVMCAVE energies before writing to the vmc.hist file. This saves both space and time. Note that the total number of moves of all electrons in a block is given by NMOVE×CORPER×NVMCAVE so you will need to reduce NMOVE proportionately if you increase NVMCAVE (this will be changed soon).

NWRCON (*Integer*) Number of configurations to be written out (VMC) or read in (VARMIN).

OPT_DTVMC (*Integer*) The purpose of OPT_DTVMC is to optimize the value of the time step DTVMC in VMC calculations in order to minimize serial correlation. This is done during the Metropolis equilibration phase. The keyword may take two possible values : OPT_DTVMC=0 (default) turns off the optimization, OPT_DTVMC=1 means DTVMC is optimized in order to achieve an acceptance ratio of (roughly) 50%. Note further that attempts have been made to design an algorithm for optimizing the timestep by directly minimizing the correlation time. In the end however, it was found that this required too many equiibration steps to get a clear minimum, and there was not much benefit over the simpler procedure of choosing a 50% acceptance ratio.

OPT_MAXEVAL (*Logical*) Maximum number of evaluations during optimization (default 200).

OPT_MAXITER (*Integer*) Largest permitted number of variance minimizer iterations (default : 150).

ORBBUF (*Logical*) Setting ORBBUF=.TRUE. turns on DMC orbital buffering. This is an efficiency device in which buffered copies of orbitals/gradients/Laplacians are kept for later reuse. This has a high memory cost. Orbital buffering should always be used unless you start running out of memory; hence the ability to turn it off.

ORB_NORM (*Real*) Allows user to change normalization of orbitals by multiplying all of them by this constant. Of course this should have no effect on the energy but it can be useful if the Slater determinant starts going singular, as it might for example for some very dilute/low density systems.

PAIR_CORR (*Logical*) Set PAIR_CORR to T to activate calculation of the pair-correlation function. Currently restricted to periodic systems. Note you also need to give values for the FIXED_E_POS block. Spherical averaging may be performed - see the PCF_SPH_MODE keyword.

PCF_SPH_MODE (*Integer*) Real space accumulation on radial bins around a fixed electron basically gives the information about the pair correlation function. When PCF_SPH_MODE has a non-zero value the accumulation on such radial bins is performed. In order to get the spherically averaged pair-correlation function, the radial distribution of 1/n(r) should be normalized by the radial distribution of unity (i.e. the bin volume). So users should first accumulate the normalized distribution with PCF_SPH_MODE=1, and then accumulate 1/n(r) with PCF_SPH_MODE=2. Theoretical background is given in the paper PRB 68, 165103 (2003).

PERMIT_DEN_SYMM (*Logical*) If this flag is set then 1) it will be assumed that if the SCF charge density expansion coefficients are real then the QMC ones are too; 2) the symmetry of the SCF charge density will be imposed upon the QMC charge density data for use in the MPC interaction. It is possible that DMC will break the symmetry of the SCF calculation; in this case the user should turn off PERMIT_DEN_SYMM.

POPRENORM (*Logical*) If poprenorm is T then in DMC after every block the number of configs (walkers) on each node will be renormalized to NCONFIG. This is achieved in subroutine RENORM using the following method: (1) each node tells node 0 how many configs it now has. (2) Node 0 instructs random configs on random nodes to be deleted or copied until the total number equals NCONFIG*NNODES. Note that it is not normally required to do this, and you should turn it on only when your run is showing abnormally large population fluctuations. Note also that in versions of CASINO prior to sometime in early 2001 this procedure was carried out automatically, but the DMC population control algorithm was then changed and the renormalization procedure became unnecessary. Note also that using this will introduce a large bias into the results. It should not be used for any serious calculations.

PRINTGSCREENING (*Logical*) Before doing a periodic Gaussian calculation, CASINO prepares lists of potentially significant (primitive) cells and sites in each such cell which could contain Gaussians having a non-zero value in a reference primitive cell centred on the origin. Zero is defined as $10^{-\mathrm{GAUTOL}}$. Turning on the PRINTGSCREENING flag prints out the important information about this screening.

QMC_DENSITY_MPC (*Logical*) If this flag is set then the QMC charge data at the end of the density.data file will be used for the MPC interaction.

RANLUXLEVEL (*Integer*) To generate the parallel streams of pseudo-random numbers for its stochastic algorithms, CASINO uses an implementation of the RANLUX algorithm. This is an advanced pseudo-random number generator based on the RCARRY algorithm proposed in 1991 by Marsaglia and Zaman. RCARRY used a subtract-and-borrow algorithm with a period on the order of 10171 but still had detectable correlations between numbers. Martin Luescher proposed the RANLUX algorithm in 1993; RANLUX generates pseudo-random numbers using RCARRY but throws away numbers to destroy correlations. RANLUX trades execution speed for quality through the choice of a 'luxury level' given in CASINO by the RANLUXLEVEL input keyword. By choosing a larger luxury setting one gets better random numbers slower. By the tests available at the time it was proposed, RANLUX at its higher settings appears to give a significant advance in quality over previous generators. The luxury setting must be in the range 0-4. Level 0 : equivalent to the original RCARRY of Marsaglia and Zaman, very long period, but fails many tests. Level 1 : considerable improvement in quality over level 0, now passes the gap test, but still fails spectral test. Level 2 : passes all known tests, but theoretically still defective. Level 3 [DEFAULT] : any theoretically possible correlations have very small chance of being observed. Level 4 : highest possible luxury, all 24 bits chaotic.

RANPRINT (*Integer*) Setting this keyword to a value greater than zero will cause the first RANPRINT numbers generated by the CASINO random number generator to be printed to a file 'random.log'. On parallel machines the numbers generated on all nodes are printed. The run scripts should pick out random.log files from different stages of a calculation (e.g. VMC config gen/DMC equil/DMC stats accumulation and rename them appropriately.

REDIST_PERIOD (*Integer*) This is the number of DMC moves between configuration redistributions on a parallel machine, where the number of configurations on each processor is equalised by transferring configurations between processors. The default is 1, but changing it to something larger could produce a performance benefit on some machines.

RELATIVISTIC (*Logical*) If RELATIVISTIC is T, then calculate relativistic corrections to the energy using perturbation theory. Note that for the moment this can only be done for closed-shell atoms. See Section 10.27 for further details.

SPARSE (*Logical*) CASINO is capable of using sparse matrix algebra in some algorithms for efficiency purposes. For systems which are definitely not sparse (orbitals not well localized) then attempting to use sparse algorithms might actually slow things down. Thus until we work out a better way you can toggle this behaviour with the SPARSE flag. In these algorithms a matrix element is considered to be zero if it less than the value of the input keyword SPARSE_THRESHOLD.

SPARSE_THRESHOLD (*Real*) CASINO sometimes uses sparse matrix algebra for efficiency purposes. This keyword defines a threshold such that a matrix element is taken to be zero if it is less than this threshold. Changing this quantity might be used to trade speed for accuracy in some cases.

SPIN_DENSITY (*Logical*) Setting SPIN_DENSITY to T will activate the accumulation of separate up- and down-spin densities in the density.data file.

TESTRUN (*Logical*) If .TRUE. then read input files, print information and stop.

TIMING_INFO (*Logical*) Setting TIMING_INFO to F (the default) will turn off the collection of subroutine timings. You might want to do this as the the timing routines can adversely affect system performance on certain computers (such as Alpha or PC clusters) especially for small systems.

TPDMC (*Integer*) TPDMC (T_p) is the number of timesteps for which the effects of changes in the (theoretically constant) reference energy should be undone in order to estimate the DMC energy at a given point. It is assumed that the best estimates of the DMC energy separated

by an amount greater than this are not correlated by fluctuations in the reference energy. Thus T_p should exceed the timescale of fluctuations in the reference energy. Umrigar suggests using $T_p = 10/\tau$ where τ is the timestep. If you set it to 9999 in the input, then the code will automatically use this value; if you set it to 0 then the reweighting scheme for population control biasing will not be used. The latter is the default. See Section 10.8.

TRIP_POPN (*Integer*) In the course of a DMC simulation, it is possible for a configuration "population explosion" to occur. If TRIP_POPN is set to 0 then nothing will be done about this. If TRIP_POPN is greater than 0 then it will attempt to restart the block if the single-node population exceeds TRIP_POPN. A general suggestion for its value would be two times NCONFIG. See the discussion in Section 10.10 for more details.

USE_COEFF_FILE (*Logical*) Use a coefficient file with the New Optimization Method. Must be set to true at the moment (default), as newopt can only cope with a varmin_coeffs.data file for the time being. (See section 10.24).

USE_NEWJAS (*Logical*) If this input parameter is set to true then CASINO will use the new form of Jastrow factor introduced around Easter 2004. If it is set to false then the old Jastrow factor employed prior to that will be used. The former requires a `jastrow.data` file, while the latter requires a `jasfun.data` file. The new Jastrow almost always gives a significantly lower VMC energy and variance than the old one.

USE_NEWOPT (*Logical*) If USE_NEWOPT is T then the coefficients of the quartic LSF will be accumulated in VMC, and the New Optimisation Method will be used in a varmin run. Note that only linear Jastrow parameters can be optimized. (See section 10.24).

USE_JASTROW (*Logical*) Use a wave function of the Slater-Jastrow form, where the Jastrow factor $\exp(J)$ is an optimizable object that multiplies the determinant part in order to intoduce correlations in the system. There are currently two forms of the Jastrow factor available in CASINO, selectable via the flag USE_NEWJAS. For historical reasons, the default value for USE_JASTROW is T, and is overriden when IRUN is 1, which is the old synonym for IRUN=2 (VMC) and USE_JASTROW=F, i.e., for a Hartree-Fock VMC (HF-VMC) calculation.

VMC_METHOD (*Integer*) VMC_METHOD selects which version of VMC to use: 1) Evaluate configuration energies at the end of the configuration move; 2) Evaluate the energy during the move by averaging electron energies over the old and new positions; or, 3) Like 1, but propose entire configuration moves instead of single-particle ones at the accept/reject stage. Method 1 is the default.

VM_DERIV_BUFFER (*Logical*) Setting this flag to T will speed up variance minimization by buffering the kinetic energy terms associated with the different factors present in the wave function. This is useful because NL2SOL spends most function evaluations in computing numerical derivatives, hence changing only one parameter at a time, which affects only a portion of the wave function. It is currently possible to buffer all five terms in the new Jastrow function, and the Slater determinant. For the old Jastrow this feature is not yet implemented.

VM_E_GUESS (*Real*) If VM_USE_E_GUESS is true then VM_E_GUESS should be supplied as an estimate of the ground-state energy.

VM_FIXNL (*Logical*) Fix nonlocal in VARMIN. Currently only .TRUE. is allowed.

VM_FORGIVING (*Logical*) Do not whinge about calculated energies not agreeing with those read in. Recommended to be set to .FALSE..

VM_INFO (*Integer*) Controls amount of information displayed during variance minimization. Takes values: 1) Display no information; 2) Display energies at each function evaluation; 3) As 2), but calculate weights as well; 4) Write out configs and their energies etc as they are read in.

VM_JASCHECK (*Logical*) Numerical check of whether value, gradient and Laplacian of Jastrow factor are consistent. Recommended to be set .TRUE..

VM_MODE (*Character*) May take values 'linear' or 'direct': see Section 10.14.
'linear' for optimizing parameters which occur linearly in J;
'direct' for optimizing parameters which occur non-linearly in J : i.e. A.

VM_OPT_DET_COEFF (*Logical*) Optimize the coefficients of the determinants in variance minimization.

VM_OPT_JASFUN (*Logical*) Optimize the Jastrow factor in variance minimization.

VM_OPT_PAIRING (*Logical*) Optimize the Rex parameter used in the electron-hole excitonic insulator phase in variance minimization.

VM_REWEIGHT (*Logical*) If .TRUE. then use reweighting in VARMIN. If .FALSE. set all weights to unity.

VM_SMOOTH_LIMITS (*Logical*) When set to T, the optimizing routine used in variance minimization is sent a smoothed version of the set of parameters. This only affects those which are to remain bounded such as Jastrow cutoffs. The result is a set of parameters which can vary in the range $(-\infty, +\infty)$, which can be more convenient than ignoring out-of-range values without the minimizer knowing. A suitable hyperbolic function is used for mapping 'limited' values into 'extended' ones and viceversa.

VM_USE_E_GUESS (*Logical*) Setting this flag to true will cause the 'variance' in varmin to be evaluated using VM_E_GUESS in place of the average energy of the config set, in an attempt to combine energy minimisation with varmin. Otherwise the least-squares function will simply be the variance of the configuration local energies.

VM_W_MIN (*Real*) Minimum value that a configuration weight may take during weighted variance minimization. This parameter should have a value between zero and one. Note that the limiting is not applied if VM_W_MAX = 0.

VM_W_MAX (*Real*) Maximum value that a configuration weight may take during weighted variance minimization. Set this to zero if you do not wish to limit the weights; otherwise it should be greater than 1.

WAVEFUNCTION *Block* The wave function block specifies the nature of the Slater part of the many-electron wave function. The xwfn.data file specifies a reference configuration, which may consist of one or more determinants. The input file allows use of either the reference configuration (GS) or to specify excitations/additions/subtractions from GS. If the excitations/additions/subtractions are made to a single determinant GS configuration then we have to specify SD in the input file and if they are made to a multi-determinant GS configuration we have to specify MD. Excitations/additions/subtractions are made using the keywords 'PR', 'PL' and 'MI,' respectively. Several changes to the GS configuration can be made at once.

WDMCMIN, WDMCMAX (*Reals*) Weighted DMC only: minimum and maximum weights. See entry for LWDMC.

7.2 Jastrow factor file: `jastrow.data`

The file that holds the parameter values for the Jastrow factor used by CASINO is called `jastrow.data`. (Note this replaces an older form of Jastrow factor called `jasfun.data` which was used until Easter 2004 - CASINO still supports this and it is therefore described in Section 10.14.4 and Appendix 19.)

The format of the `jastrow.data` file is as follows. This represents the state of the file *before* optimization since the variable parameters are not given explicitly and are therefore assumed by CASINO to be zero.

```
START JASTROW
Title
Title of system goes here.
Truncation order
  3
START U TERM
Number of sets
  1
START SET 1
Spherical harmonic l,m
  0 0
Expansion order
  6
Spin-dep params (0->uu=dd=ud; 1->uu=dd/=ud; 2->uu/=dd/=ud)
  1
Cutoff (a.u.)     ;  Optimizable (0=NO; 1=YES)
  0.d0                            1
Parameter values  ;  Optimizable (0=NO; 1=YES)
END SET 1
END U TERM
START CHI TERM
Number of sets
  2
START SET 1
Spherical harmonic l,m
  0 0
Number of atoms in set
  1
Labels of the atoms in this set
  1
Impose electron-nucleus cusp (0=NO; 1=YES)
  0
Expansion order
  6
Spin-dep params (0->u=d; 1->u/=d)
  0
Cutoff (a.u.)     ;  Optimizable (0=NO; 1=YES)
  0.d0                            0
Parameter values  ;  Optimizable (0=NO; 1=YES)
END SET 1
START SET 2
Spherical harmonic l,m
  0 0
Number of atoms in set
  4
Labels of the atoms in this set
  2 3 4 5
Impose electron-nucleus cusp (0=NO; 1=YES)
  0
Expansion order
```

```
  6
Spin-dep params (0->u=d; 1->u/=d)
  0
Cutoff (a.u.)     ;  Optimizable (0=NO; 1=YES)
  0.d0                            0
Parameter values  ;  Optimizable (0=NO; 1=YES)
END SET 2
END CHI TERM
START F TERM
Number of sets
  2
START SET 1
Number of atoms in set
  1
Labels of the atoms in this set
  1
Prevent duplication of u term (0=NO; 1=YES)
  1
Prevent duplication of chi term (0=NO; 1=YES)
  1
Electron-nucleus expansion order
  2
Electron-electron expansion order
  2
Spin-dep params (0->uu=dd=ud; 1->uu=dd/=ud; 2->uu/=dd/=ud)
  0
Cutoff (a.u.)     ;  Optimizable (0=NO; 1=YES)
  0.d0                            0
Parameter values  ;  Optimizable (0=NO; 1=YES)
END SET 1
START SET 2
Number of atoms in set
  4
Labels of the atoms in this set
  2 3 4 5
Prevent duplication of u term (0=NO; 1=YES)
  1
Prevent duplication of chi term (0=NO; 1=YES)
  1
Electron-nucleus expansion order
  2
Electron-electron expansion order
  2
Spin-dep params (0->uu=dd=ud; 1->uu=dd/=ud; 2->uu/=dd/=ud)
  0
Cutoff (a.u.)     ;  Optimizable (0=NO; 1=YES)
  0.d0                            0
Parameter values  ;  Optimizable (0=NO; 1=YES)
END SET 2
END F TERM
END JASTROW
```

There are two additional terms which may be of benefit in periodic systems but which in practice are very rarely used (insert these between END F TERM and END JASTROW above).

```
START P TERM
Spin dep (0->uu=dd=ud; 1->uu=dd/=ud; 2->uu/=dd/=ud)
  1
Number of simulation-cell G-vectors (NB, cannot have both G & -G)
  6
G-vector (in terms of rec latt vects) ; label
     0      0     -1   1
     0     -1      0   1
    -1      0      0   1
     1      1      1   1
     1      1      0   2
     1      0      1   2
Parameter value ; Optimizable (0=NO; 1=YES)
END P TERM
START Q TERM
Spin dep (0->u=d; 1->u/=d)
  0
Number of primitive-cell G-vectors (NB, cannot have both G & -G)
  10
G-vector (in terms of rec latt vects) ; label
  0  -1  1      -2
  -1  1  0      -2
  -1  0  1      -2
  1  1  1      -1
  -1  0  0      1
  0  0  -1      1
  0  -1  0      1
  1  1  2      2
  2  1  1      2
  1  2  1      2
Parameter value ; Optimizable (0=NO; 1=YES)
END Q TERM
END JASTROW
```

Notes:

- There are 5 types of term available: u (isotropic electron–electron terms), χ (isotropic electron–nucleus terms), f (isotropic electron–electron–nucleus terms), p (plane-wave expansions in electron–electron separation) and q (plane-wave expansions in electron position). All of these terms are optional: e.g., omitting all the lines from "START Q TERM" to "END Q TERM" will give a Jastrow factor with no q-terms. In many cases (particularly in the presence of pseudopotentials) only u and χ terms are needed.

- The 'truncation order' should be either 2 or 3. If it is of value C then then the Jastrow factor is C times differentiable everywhere; it must therefore be at least 2 for the kinetic-energy integrand to be well-defined. If it is 3 then the local energy is continuous in configuration space (assuming the orbitals are smooth). This is not strictly required, and leads to the loss of some variational freedom, but it makes the numerical optimization of cutoff lengths easier.

- In a future release, the u and χ terms may be made anisotropic; there are placeholders for this in `jastrow.data`. At present, however, both must be isotropic. There should only be one set of u terms, and the spherical harmonic l and m values should therefore be set to 0.

- The 'expansion order'of u determines the number of parameters. Typically it is given a value between 4 and 8.

- The 'spin-dep params' line allows the user to specify whether the same u-parameters are to be used for parallel- and antiparallel-spin electron-pairs. If the value is set to 0 then the same parameters are used for parallel and antiparallel pairs (this option should *not* be used in general);

if the value is set to 1 then different parameters are used for parallel and antiparallel pairs; if it is set to 2 then different parameters are used for up-spin, down-spin and opposite-spin pairs: this is useful for spin-polarized systems.[5]

- The cutoff length is given a default value if it is set to 0. Note that it is possible to optimize the cutoff length using variance minimization. This can be useful, because the choice of cutoff length has a significant effect on the optimized energy and variance. Unfortunately, optimizing the cutoff length is numerically difficult: variance minimization will take many more iterations to converge if the cutoff is optimizable. Setting the truncation order to 3 can help somewhat.

- It is possible to specify exactly which of the expansion coefficients ('parameter values') are optimizable and which are not. One does not need to specify all of the expansion coefficients: any that are not listed in `jastrow.data` are assumed to be zero. Furthermore, if too many are given, then the extra parameters will be ignored. If all of the parameter values in all of the Jastrow terms are blank, as is the case in the example given above, then only the Slater wave function will be used for the first configuration generation run when performing variance minimization.[6]

- Different χ-functions are used for different species of ion: the 'number of sets' should be chosen to be equal to the number of chemically-distinct species.

- The ions in each set are specified by giving a list of the numbers that label them: these are the same as the labels used in the `xwfn.data` file that specifies the geometry of the system.

- It is possible to make the Jastrow factor enforce the electron–nucleus cusp condition. This should only be done if the χ-set contains bare nuclei and the orbitals do not satisfy the cusp condition. With a Gaussian basis set, it is much better to use the in-built cusp correction algorithm activated with the input keyword CUSP_CORRECTION rather than use the Jastrow.

- There are two spin-dependence options for χ: if it is 0 then the same parameters are used for up- and down-spin electrons; if it is 1 then different parameters are used.[7]

- Similar comments to those made for u apply to the 'spherical harmonic' labels, the cutoff length and the parameter values of χ.[8]

- The f function contains terms that approximately duplicate the u and χ terms: additional constraints can be placed on f to remove these terms if desired. This is usually a good idea if the cutoff length for f is about half that of u and about the same as that of the corresponding χ-set. If the cutoff length of f is very different then duplication of u and χ should be permitted.

- The number of f-parameters grows very rapidly with the electron–electron and electron–nucleus expansion orders. These should normally be either 2 or 3.

- The spin-dependence options for f are exactly the same as for u.

- The p and q terms can only be present in periodic systems. In practice, neither appears to be particularly useful. Note that q should only be used if the origin is a centre of inversion symmetry of the charge density.

- The spin-dependence options for p and q are the same as those of u and χ respectively.

[5] For electron–hole systems, a 'spin-dep params' value of 0 means that the same coefficients are used for all spin/particle types; 1 means that different parameters are used for electron–electron, electron–hole and hole–hole pairs; 2 means that different parameters are used for parallel and antiparallel electron and hole pairs, but that electron–hole pairs all have the same parameters; 3 means that different parameters are used for parallel and antiparallel spin-pair types and different parameters are used for each type of particle-pair.

[6] The u-parameters are listed in the following order for electron systems: coefficients for spin-up pairs; coefficients for antiparallel pairs (if spin-dependence is 1 or 2); coefficients for spin-down pairs (if spin-dependence is 2). For each spin-pair, the number of parameters is equal to the expansion order.

[7] For electron–hole systems, a spin-dependence value of 0 means that the same parameters are used for all spins and particles; 1 means that different parameters are used for electrons and holes.

[8] For electron systems, the parameter values are given for spin-up electrons, then (if the spin-dependence is 1) for spin-down electrons. For each spin-type, the number of parameters is equal to the expansion order.

- For p, a list of simulation-cell **G**-vectors must be provided. These are specified in terms of the reciprocal-lattice vectors. Only one out of each **G** and $-\mathbf{G}$ should be specified. The same parameter value is used for **G**-vectors with the same label.
- For q, a list of primitive-cell **G**-vectors must be provided. Again, **G**-vectors with the same label have the same parameter value. It is possible to specify a negative relationship between parameter values by using a negative label for the appropriate **G**-vectors.

7.3 Pseudopotential file : xx_pp.data

CASINO can carry out all-electron or pseudopotential calculations, but it is normally advantageous to replace the core electrons by a pseudopotential. CASINO will automatically treat an atom as all-electron unless there exists a pseudopotential file **xx_pp.data** where **xx** is the symbol for the element in question in lower case. This file contains the different angular momentum components of the pseudopotential given on a radial grid and some auxiliary information in the following format.

```
LSDA Pseudopotential in real space for Si
Atomic number and pseudo-charge
14 4d0
Energy units (rydberg/hartree/ev):
rydberg
Angular momentum of local component (0=s,1=p,2=d..)
1
NLRULE override (1) VMC/DMC (2) config gen (0 ==> input/default value)
0 0
 Number of grid points
        2476
 R(i) in atomic units
  0.000000000000000E+000
  7.178690774405650E-012
.
   39.6567798705125
   40.0553371342383
 r*potential (L=0) in Ry
     0.000000000000000E+00
     0.131424063731862E-10
.
     -8.00000000000000
     -8.00000000000000
 r*potential (L=1) in Ry
     0.000000000000000E+00
    -0.574393968349227E-10
.
     -8.00000000000000
     -8.00000000000000
 r*potential (L=2) in Ry
     0.000000000000000E+00
    -0.128711823920867E-09
.
     -8.00000000000000
     -8.00000000000000
Core polarization terms
0.1650 0.5446        ! alpha (a.u.) rbaree (a.u.) usually set to 0.5*(rbars+rbarp)
0.5216 0.5676 0.7172 ! rbar for s,p,d  (a.u.)
```

The NRULE parameters control the grid on which the angular integration is performed, see Section 10.18, and are normally specified in the file 'input'. The values given in 'input' are the default for all atoms in the system but they can be overridden for particular elements by setting the parameters in this file to non-zero values. Unless you explicitly wish to use a core polarization potential (see Section 10.19), the few lines at the bottom relating to this should be omitted.

CASINO pseudopotential library: www.tcm.phy.cam.ac.uk/~mdt26/casino_users.html

7.4 Charge density file : density.data

This contains the Fourier transform of the charge density, and it is used in the evaluation of the Modified Periodic Coulomb (MPC) interaction. Before doing a QMC calculation with the MPC interaction, this file should be generated from the trial wave function given in the [x]wfn.data file by using 'runvmc' to run CASINO with IRUN set to 6 in the input file (note that the 'eepot.data' file is also required for this). Note that if the IDEN parameter is set to 1 in the input file, then the QMC density will be accumulated at the end of the file following the description of the charge density of the trial wave function.

```
 Plutonium-doped lanthanum nickelate
 Charge density in reciprocal space
 Real space translation vectors (Cartesians in atomic units)
   0.000000000000    3.370326931161    3.370326931161  ! a1
   3.370326931161    0.000000000000    3.370326931161  ! a2
   3.370326931161    3.370326931161    0.000000000000  ! a3
 Reciprocal space translation vectors (Cartesians in atomic units)
  -0.932132911066    0.932132911066    0.932132911066  ! b1
   0.932132911066   -0.932132911066    0.932132911066  ! b2
   0.932132911066    0.932132911066   -0.932132911066  ! b3
 Number of atoms in the unit cell
           2                                          ! # of atoms in cell
  Positions of atoms (Cartesians in atomic units)
  57    0.8425817327    0.8425817327    0.8425817327    ! Atomic # and position
  6   -0.8425817327   -0.8425817327   -0.8425817327    ! Atomic # and position
 Number of G-vectors
        6861                                          ! Number of G-vectors
  G-vectors (Cartesians in atomic units)
   0.000000000000    0.000000000000    0.000000000000  ! List of G-vectors
.
.
 -13.049860754934    0.000000000000   13.049860754934
 Charge density from SC calculation
   8.00000000000000                                   ! List of rho(G)
.
.
 -2.020855523387648E-006
 BOOLEAN for QMC charge density to follow
 .TRUE.                          ! BOOLEAN for QMC charge density
           8                     ! # of primitive cells in simulation cell
   2.99883837683579              ! # of moves
          32                     ! Number of up-spin electrons
          32                     ! Number of down-spin electrons
        6861                     ! Number of G-vectors
   191.925656117491              ! List of QMC rho(G). To get the correct
.                                ! correct normalization divide by the #
.                                ! of moves and number of primitive cells
   9.16791786818292
```

Notes:

1. The utility 'd2rs' can be used to convert the charge density in reciprocal space into a real space density on a grid. This can be converted into a format readable by 'gnuplot' using the 'plot_density' utility.

7.5 MPC file : eepot.data

This contains the Fourier transform of the function $f(\mathbf{r})$ defined in Section 10.20.4. It is required when using the Modified Periodic Coulomb (MPC) interaction to compute the electron-electron interactions. Before doing a QMC calculation with the MPC interaction, this file should be generated from the trial wave function given in the [x]wfn.data file by using 'runvmc' to run CASINO with IRUN set to 5 in the input file (Note that the 'density.data' file is also required for this.).

The format of the file should be obvious from looking at some of the examples.
NB: it is not necessary to regenerate this file for different supercell sizes, provided the lattice vectors in eepot.data are obtainable from the lattice vectors in (x)wfn.data through scaling by a single parameter.

7.6 Trial wave function files : awfn.data, bwfn.data, gwfn.data, pwfn.data, swfn.data

These files contain the data defining the determinant part of the trial wave function. The data can be given in a Gaussian basis set (gwfn.data), in a plane wave basis set (pwfn.data), in a blip function basis set (bwfn.data) or in splines (swfn.data). The awfn.data file contains a trial wave function for an atom with the orbitals given explicitly on a radial grid.

Without going into details of specific formats, the files basically contain the following information:

- Basic info about the trial wave function generating calculation (e.g., DFT/HF/etc.) including total energy and components.
- Geometry of the system.
- Details of the k-space net used by the program that generated the trial wave function (only if the systems is periodic).
- Details of the basis set :

  - Exponents, contraction coefficients, shell types (Gaussians)
  - G-vectors of the plane waves
  - Blip grid
- Multi-determinant properties of the trial wave function (if any).
- Specification of the orbitals:

  - Gaussian coefficients
  - Plane wave coefficients
  - Blip coefficients
- Eigenvalue spectrum (used to work out which orbitals to occupy and, crudely, whether the thing is a metal or an insulator).
- The output file from the program that generated the trial wave function (not read by CASINO - for reference only).

These files are generated automatically by various utilities available for different electronic structure programs (see Sections 12, 13, 15, 16 and 16). The format of the different files should be clear from looking at the various examples for different dimensionalities and basis sets.

7.6.1 gwfn.data file specification

```
Dimensions of allocated arrays are shown.

For the larger fields I use the following formatting.

 % strings (title,code,method,functional) may be up to 80 characters long.
 * free format
 $ FORMAT(3(1PE20.13)) - for reals naturally grouped in triples (e.g. coords)
 & FORMAT(4(1PE20.13)) - other reals (e.g. gaussian exponents)
 @ FORMAT(8I10)        - lengthy lists of integers

use e.g. write(IO,format=&)(array(i),i=1,size)

First line of file is TITLE field.

MDT 1997

--------------------------------------------------------------------------------

% TITLE (the title)

BASIC INFO
----------
% CODE:
  name of code producing this file
  (i.e. CRYSTAL95, CRYSTAL98, GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, TURBOMOLE)
% METHOD:
  Comment - RHF/ROHF/UHF/DFT/S-DFT/CI/etc.
% FUNCTIONAL
  DFT functional name. If method not DFT then 'none'.
* PERIODICITY:
  system dimension 0,1,2,3 => molecule, polymer, slab, solid
* SPIN_UNRESTRICTED:
  .true. or .false. (i.e. different orbitals for different spins)
* EIONION:
  nuclear-nuclear repulsion energy (au/atom)
* NUM_ELECTRONS:
  number of electrons per primitive cell

GEOMETRY
--------
* NUM_ATOMS:
  number of atoms per primitive cell
$ BASIS(3,NUM_ATOMS):
  atomic positions(au)
@ ATNO(NUM_ATOMS):
  atomic numbers for each atom
& VALENCE_CHARGE(NUM_ATOMS):
  valence charges for each atom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PERIODIC INSERT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
$ LATTICE_VECTORS(3,3):
  primitive lattice vectors (au)

K SPACE NET
-----------
```

```
* NUM_K:
  no. of k points in BZ
* NUM_REAL_K:
  no. of 'real' k points
  (all components of 'real' k points are either zero or half a
   reciprocal lattice vector)
$ KVEC(3,NUM_K):
  k point coordinates (a.u.)
  NB: coordinates of 'real' k points must occupy the first num_real_k
  positions in kvec.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END PERIODIC INSERT%%%%%%%%%%%%%%%%%%%%%%%%%

BASIS SET
---------
* NUM_CENTRES
  number of centres with associated Gaussian functions
  (i.e. number of atoms + number of non-atom Gaussian sites) per primitive cell
* NUM_SHELLS:
  number of shells per primitive cell
* NUM_AO:
  number of basis functions ('AO') per primitive cell
* NUM_PRIMS:
  number of Gaussian primitives per primitive cell
* HIGHEST_ANG_MOM:
  highest angular momentum of shells
@ SHELL_AM(NUM_SHELLS):
  code for shell type
  s=1, sp=2, p=3, d= 4, f= 5 etc.. (harmonic representation)
                  d=-4, f=-5       (cartesian representation)
@ NUMPRIMS_IN_SHELL(NUM_SHELLS):
  Number of primitive Gaussians in each shell
@ FIRST_SHELL(NUM_CENTRES+1)
  Sequence number of first shell on each Gaussian centre.
  Allows e.g.
  do n=1,num_centres
   do shell=first_shell(n),first_shell(n+1)-1
    blah..
   enddo
  enddo
  to loop over shells on each centre
  Note dimension.
& EXPONENT(NUM_PRIMS):
  exponents of Gaussian primitives
& C_PRIM(NUM_PRIMS)
  correctly normalized contraction coefficients
& C_PRIM2(NUM_PRIMS) **must be omitted if no sp shells in basis**
  correctly normalized 2nd contraction coefficients
  (i.e. p coefficient for sp shells, zero otherwise)
$ SHELL_POS(3,NUM_SHELLS)
  positions of shells (not necessarily atom-centred)

MULTIDETERMINANT INFORMATION
----------------------------

% GS - Ground state calculation

or
```

```
% SD - Single determinant inc. excitations/additions/subtractions

  Example:
         SD                    Single det calculation
         DET 1 1 PR 2 1 5 1    Promote electron in band 2 kpoint 1
                               to band 5 kpoint 1 in determinant 1,
                               spin 1 ("up")

         or

         DET 1 2 PL 6 1        Add electron spin 2 ("down") to det 1,
                               band 6 kpoint 1

         or

         DET 1 2 MI 6 1        Remove the added electron

or

% MD - Multiple determinants
         MD                    Multideterminant
         3                     3 determinants
         1.d0                  Determinant 1 prefactor
         2.d0                  Determinant 2 prefactor
        -1.d0                  Determinant 3 prefactor
         DET 3 1 PR 2 1 5 1    Promotion as examples above


ORBITAL COEFFICIENTS
--------------------

& CK(NUM_REAL_K*NUM_AO*NUM_AO + NUM_COMPLEX_K*NUM_AO*NUM_AO)
  block as follows
   ----spin (spin-polarized calcs only)---------------------------------
  /                                                                      \
   ----k (for solids)-----------------------------
  /                                              \
   ----bands-(solids)_or MOs (molecules)------
  /                                           \
   -AO-basis functions grouped by shell-
  /                                     \

  Complex coefficients are given as 2 adjacent real numbers (real,imaginary).
  Ordering of d orbital coefficients;
  z2, xz, yz, x2-y2, xy
  Ordering of f,g,h...: m=0,1,-1,2,-2,3,-3,4,-4.....

OTHER STUFF
-----------

* Anything you like can be written here - intended for copy of input/output
  files from DFT/HF/etc.. program.
```

7.6.2 pwfn.data file specification

Orbitals expanded in plane waves.
pwfn.data file example (should be self explanatory):

```
 Si diamond

 BASIC INFO
 ----------
 Generated by:
  CASTEP
 Method:
  DFT
 DFT Functional
  LDA
 Pseudopotential
  LDA Trouiller-Martin (1551 coeff)
 Plane wave cutoff (au)
  7.5
 Spin polarized:
  F
 Total energy (au per primitive cell)
  -7.84635517057440
 Kinetic energy (au per primitive cell)
    3.24780615624099
 Local potential energy (au per primitive cell)
  -1.03508540683458
 Non-local potential energy (au per primitive cell)
  0.144898714904598
 Electron-electron energy (au per primitive cell)
  -1.80295658630558
 Ion-ion energy (au per primitive cell)
  -8.40101804857984
 Number of electrons per primitive cell
    8

 GEOMETRY
 --------
 Number of atoms per primitive cell
  2
 Atomic numbers and positions of atoms (au)
  14   1.2824155389173550   1.2824155389173550   1.2824155389173550
  14  -1.2824155389173550  -1.2824155389173550  -1.2824155389173550
 Primitive lattice vectors (au)
  0.000000000000000E+000   5.12966215566942        5.12966215566942
    5.12966215566942       0.000000000000000E+000   5.12966215566942
    5.12966215566942        5.12966215566942       0.000000000000000E+000

 G VECTORS
 ---------
 Number of G-vectors
         341
 Gx Gy Gz (au)
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000
<snip>
    2.44974624702536        2.44974624702536        2.44974624702536
```

```
 WAVE FUNCTION
 -------------
 Number of k-points
   8
 k-point # ; # of bands (up spin/down spin) ; k-point coords (au)
   1 4 0 0.0 0.0 0.0
 Band, spin, eigenvalue (au)
   1 1 -2.432668987064920E-002
 Eigenvector coeficients
 -0.953978956601082
<snip>
  0.000000000000000E+000
 Band, spin, eigenvalue (au)
   2 1 0.417639684278199
 Eigenvector coeficients
 -1.576227515378431E-012
<snip>
  0.000000000000000E+000
 Band, spin, eigenvalue (au)
   3 1 0.417639711691157
 Eigenvector coeficients
  6.348405363267598E-008
<snip>
  0.000000000000000E+000
 Band, spin, eigenvalue (au)
   4 1 0.417639745897256
 Eigenvector coeficients
 -2.030042411030856E-009
<snip>
  0.000000000000000E+000
 k-point # ; # of bands (up spin/down spin) ; k-point coords (au)
   2 4 0 -0.3062182808781698 -0.3062182808781698 0.3062182808781698
 Band, spin, eigenvalue (au)
   1 1 6.261478573034787E-002
 Eigenvector coeficients
  0.683590846620266
<snip>
  0.000000000000000E+000
 Band, spin, eigenvalue (au)
           2           1  0.156856931414136
etc.. for all 8 k points with 4 bands per k point in this case.

OTHER STUFF
-----------

* Anything you like can be written here - intended for copy of input/output
  files from the PW DFT program.
```

7.6.3 bwfn.data file specification

Orbitals expanded in blip functions. An example of a bwfn.data file for bulk silicon is given below:

```
bulk Si

BASIC INFO
----------
```

```
Generated by:
PWSCF
Method:
DFT
DFT Functional:
unknown
Pseudopotential
unknown
Plane wave cutoff (au)
 7.5
Spin polarized:
 F
Total energy (au per primitive cell)
 -62.76695352625196
Kinetic energy (au per primitive cell)
 0.E+0
Local potential energy (au per primitive cell)
 0.E+0
Non local potential energy(au per primitive cell)
 0.E+0
Electron electron energy (au per primitive cell)
 0.E+0
Ion ion energy (au per primitive cell)
 -67.208144397949283
Number of electrons per primitive cell
 64

GEOMETRY
--------
Number of atoms per primitive cell
 16
Atomic number and position of the atoms(au)
    14     -1.282415538750     -1.282415538750     -1.282415538750
    14     -1.282415538750      3.847246616250      3.847246616250
     :
     :
    14     11.541739848750      6.412077693750      6.412077693750
    14     11.541739848750     11.541739848750     11.541739848750
Primitive lattice vectors (au)
     10.259324310000     10.259324310000      0.000000000000
      0.000000000000     10.259324310000     10.259324310000
     10.259324310000      0.000000000000     10.259324310000

G VECTORS
---------
Number of G-vectors
 2085
Gx Gy Gz (au)
      0.000000000000      0.000000000000      0.000000000000
     -0.306218280918     -0.306218280918     -0.306218280918
     -0.306218280918     -0.306218280918      0.306218280918
         :
         :
      2.143527966427      2.755964528263      1.531091404591
      1.531091404591      3.368401090099      0.918654842754
Blip grid
  20  20  20
```

```
WAVE FUNCTION
-------------
Number of k-points
 1
k-point # ; # of bands (up spin/down spin) ; k-point coords (au)
   1  32   0  0.0000000000000000  0.0000000000000000  0.0000000000000000
Band, spin, eigenvalue (au)
    1    1     -0.015443213499
Blip coefficients
     -0.347534087942
     -0.517756216863
     -0.528529751776
      :
      :
     -0.450993755155
      0.024823593723
     -0.022992059309
Band, spin, eigenvalue (au)
    2    1     -0.009057759900
Blip coefficients
     -0.097602672632
     -0.176031268912
     -0.188226226118
      :
      :
    (for 32 bands)
```

7.6.4 swfn.data file specification

Orbitals expanded in spline functions. Soon.

7.6.5 awfn.data file specification

Atomic orbitals given on a radial grid. Below is an example of an awfn.data file for beryllium:

```
Atomic Be wave function in real space
Atomic number
 4
Total number of orbitals
 2
The 1s(2)2s(2) [1S] state electronic configuration
Number of up, down spin electrons
 2  2
States
1  1  0  0                 % label of spin up electron, quantum number n, l, m
2  2  0  0
1  1  0  0                 % label of spin down electron, n, l, m
2  2  0  0
Radial grid (a.u.)
   301                     % Number of radial grid points are given
    0.000000000000000E+00  % Distance from centre of atom r
    0.457890972218354E-02
    0.477372816593466E-02
    0.497683553179352E-02
    0.518858448775392E-02
    0.540934270674177E-02
    0.563949350502860E-02
```

```
     :
     :
    0.108431746952108E+04
    0.113045182349640E+04
    0.117854905151603E+04
Orbital # 1 [1s]
 0 1 0                        % spin type [0=unpolarized, 1=up, 2=down], n, l
    0.000000000000000E+00     % r * Value of trial wave function at point r
    0.659528705936585E-01
    0.687054582044431E-01
    0.715705591983797E-01
     :
     :
    0.000000000000000E+00
    0.000000000000000E+00
    0.000000000000000E+00
Orbital # 2 [2s]
 0 2 0
    0.000000000000000E+00
    0.120187930416515E-01
    0.125203784849961E-01
    0.130424628552784E-01
     :
     :
    0.000000000000000E+00
    0.000000000000000E+00
    0.000000000000000E+00
    0.000000000000000E+00
```

7.7 heg.data file

The heg.data file is used to define the trial wave function used in calculations of electron and electron-hole systems. It is the equivalent of the [x]wfn.data file used for calculations of molecular/crystalline systems, but it is entirely self-contained and does not need to be generated by an external program. See CASINO/examples/electron_phases and CASINO/examples/electron_hole_ phases for practical examples with different wave function types (2D/3D fluid/crystal/excitonic insulator).

(NB: in the case of the jellium slab system (ETYPE=6) the heg.data file takes on quite a different form - see the next section).

The basic data required are the dimensionality (2D or 3D), the Wigner-Seitz radius parameter r_s and the translation vectors of the simulation cell. CASINO works out the scaling of the translation vectors from the number of electrons and the value of r_s and builds the determinant part of the ground state wave function accordingly.

Running through the parameters in the remainder of the heg.data file:

- Exciton units may be selected for calculations of electron-hole phases.
- The mass values only affect electron-hole phases and the logical flag indicates whether to use particle symmetries when optimizing the Jastrow factor. Only set this true when the masses are equal.
- In 2D, you may separate up- and down-spin electrons for electron-only phases or electrons and holes for electron-hole phases. This is equivalent to having two 2D planes dz apart.
- For excitonic-insulator phases of the electron-hole system you may set the orbital parameter $R_{\rm ex}$ in $\phi = \exp(-r/R_{\rm ex})$. Alternatively you may choose to use a linear combination of Gaussian orbitals. If $N_g > 0$ the Gaussian form will be selected and should be set to one unless you wish to fit the coefficients to the exponential form. Note, by default you should include all contributions of orbitals from other cells into the zero cell when dealing with pairing wave functions.
- For crystalline phases, you may select from a set of predefined crystal structures or define your own by selecting 'manual'. If there is a separation along the z-axis in 2D, as mentioned earlier, the crystal structure in the two planes must be the same, though an offset in the x- and y-direction may be defined. The localized orbitals are defined as Padé functions:

  $$\Phi_i(\mathbf{r}) = \exp\left(-\frac{k_1\,|\mathbf{r}-\mathbf{R}_i|^2}{1+k_2\,|\mathbf{r}-\mathbf{R}_i|}\right),$$

  where the ith orbital is centred at $\mathbf{R}_i$. The (positive) Padé coefficients k_1 and k_2 are defined next in the input file and you may select to optimize them all or constrain them due to particle symmetry.

  The final section of the file defines the primitive crystal lattice. For the same structure as the supercell you would select one basis function and centre it on the origin. To define a different structure, say BCC primitive with a cubic supercell, you would define the appropriate number and position of basis functions. The positions written in terms of fractions [0,1] of the supercell vectors. The spin variable defines a relative spin orientation between sites, assuming you have an antiferromagnetic phase. Depending on which lattice you select, you will be allowed to simulate different numbers of electrons, one per site.
- Note that the one-particle density matrix and relativistic options exist only in test programs external to CASINO and have not yet been re-implemented in the main source. The program will stop if you turn on these flags.
- Some excited state calculations can be performed—control these through the wave_function block in the input file.

```
QMC DATA FILE FOR HOMOGENEOUS ELECTRON and ELECTRON-HOLE PHASES
---------------------------------------------------------------

Title
-----
Electron gas (3D ferromagnetic fcc Wigner crystal)

Dimensionality
--------------
 3

r_s parameter (==> density) ; flag .true. if r_s given in exciton units
-----------------------------------------------------------------------
 1.0 .false.

Simulation cell lattice vector (unscaled) 3x3 for 3D system, 2x2 for 2D system
-----------------------------------------------------------------------------
 0.0 0.5 0.5
 0.5 0.0 0.5
 0.5 0.5 0.0

Electron mass; Hole mass; ee==hh flag in Jastrow (relevant only in e-h systems)
-------------------------------------------------------------------------------
 1.0 1.0 .true.

Flag to separate layers; z-axis separation of layers (au) (relevant only in 2D)
-------------------------------------------------------------------------------
.false.  0.0

Pairing wave function parameter, Rex (relevant only for e-h gas with ETYPE=4)
-----------------------------------------------------------------------------
 1.0
Include all cells contributing to wave function?
.true.
Number of gaussians Ng (ndet=2 only)
Ng lines : coefficients a_i, Gaussian exponents b_i
 1
1.0 2.0
Fit Gaussian parameters to exp[-r] (.true.) or allow to vary freely (.false.)
 .false.

Accumulate pair correlation function ; number of bins
-----------------------------------------------------
.false. 5000

Accumulate 1 particle density matrix
------------------------------------
.false.

Turn on relativistic effects
----------------------------
.false.

Crystalline phase data (relevant only for e or e-h system with ETYPE=2 or 5)
-----------------------------------------------------------------------------
Type of crystal
3D:(C)ubic,(B)cc,(F)cc | 2D:(R)ectangular,(H)exagonal,(T)riangular | manual (M)
```

```
 F
Ferromagnetic (F) or (if not frustrated) antiferromagnetic (A)
 F
Offset (x/y) between lattices in each layer (relevant only for 2D systems)
 0.d0 0.d0
optimize all parameters
 .true.
Pade coefficients k1,k2 for (in order)up elecs;down elecs;(up holes;down holes)
  0.354388900376931       0.412126133902296
  0.706867779971532       0.287471090722791
Number of localized orbitals
 1
Positions of localized wave functions and spin of associated particle
c1*a1+c2*a2+c3*a3 , spin
  0.d0  0.d0  0.d0  1
```

7.8 Jellium slabs : heg.data file with ETYPE=6.

When ETYPE=6, the heg.data file format is entirely different from that of the various homogeneous electron (or hole) systems available. The simulation cell size (in the x and y directions) is determined by a combination of the density, given here by the r_s parameter, the number of electrons, and the slab width. The slab width is the extent in the z- direction of the region containing the positive background charge. The z-dependency of the wave function is described using Fourier sine series within a region determined by the wave function width; outside this region, it is assumed to be zero. Sub-bands are allowed, and a set of Fourier coefficients must be supplied for each; only one set of wavevectors is used. The bands may be filled automatically by setting the 'Occupy by energy' option. Alternatively, the band occupation for each spin may be specified directly. The energy width of each band is given next. It is used when occupying bands automatically, and also as a consistency check. The set of wave vectors k is given next, followed by the set of Fourier coefficients $C(k)$ for each band. The final z-dependence of the wave function is the same as within the Jastrow factor (shown above).

```
QMC data file for jellium slab
------------------------------

r_s (density parameter)
-----------------------
   2.00000000000000

Slab width, wave function width
-------------------------------
   20.0000000000000        102.400000000000

Number of bands, number of coefficients for Fourier sine series
---------------------------------------------------------------
           7        1024

Occupation by energy
--------------------
 F

Band occupation
---------------
          29          29
          25          25
          21          21
          21          21
          13          13
           9           9
           1           1

Width of each subband
---------------------
  0.519117510000000
  0.488868040000000
  0.436931490000000
  0.364359310000000
  0.280391730000000
  0.184220470000000
  8.550147999999999E-002

Wavevectors in z-direction
--------------------------
```

```
              0.0306796157577128
              0.0613592315154256
                .
                .
                .
             31.3852469201402166
             31.4159265358979294

Coefficients
------------
Band  1
--------
              0.0260839121091309
              0.0000000000000000
                .
                .
                .
              0.0000000000044777
              0.0000000000000000
Band  2
--------
              0.0000000000000002
              0.0089256999176878
                .
                .
                .

etc.
```

7.9 vmc.hist file

This file records information about each block of a VMC run. The first line gives the average of the local energy over the block and its standard error in Hartrees per primitive cell (per electron for HEG). The second line contains six integers and one energy:

1. Number of moves in the block at which the energy is evaluated (NMOVE).
2. Number of ions.
3. Number of up-spin electrons.
4. Number of down-spin electrons.
5. Periodicity: the number of dimensions in which the system is periodic (0–3).
6. Number of energy terms in file to average (needed by the REBLOCK utility). There are fifteen energy terms at present (see below).
7. The ion-ion Coulomb interaction energy per ion.

Below these, for each of the moves in the block, rows containing fifteen energy terms are given:

1. Local energy calculated using the $1/r$ or Ewald Coulomb interaction energy.
2. Total potential energy.
3. KE_i. Expectation of the kinetic energy operator (see Section 10.17).
4. T_i. (See Section 10.17).
5. F_i^2. Square modulus of the drift vector (see Section 10.17).
6. Electron-electron interaction energy ($1/r$, Ewald or MPC).
7. Electron-ion interaction energy.
8. Non-local energy.
9. MPC energy.
10. Short range part of MPC.
11. Long range part of MPC.
12. Electron-ion CPP term.
13. Electron part of CPP term.
14. Electron-electron part of CPP.
15. Average of the local energy calculated with the MPC.

All energies are quoted in Hartrees per primitive cell (per electron for electron phases, per particle for electron-hole phases). The REBLOCK utility should be used to analyse vmc.hist files and change energy units if desired. Information about the different components of the energy and how they are calculated may be found in Sections 10.16–10.20.4.

7.10 dmc.hist and dmc.hist2 files

The file dmc.hist contains the most important information from a DMC run, while the dmc.hist2 file contains less important information. There are 13 columns of data in the dmc.hist file

1. Time step counting from the beginning of the run.
2. Number of configurations at this time step.
3. Local energy averaged over the configurations.
4. Reference energy.
5. Best estimate of ground-state energy.
6. Acceptance ratio at this time step.
7. Number of the step within the current block.
8. Block number.
9. Ratio of the number of configurations to the target number.
10. Number of ions.
11. Periodicity: the number of dimensions in which the system is periodic (0–3).
12. Growth estimator of electron energy.
13. Total weight of all configurations [NEW 1/2003].

The dmc.hist2 file contains seventeen columns of data:

1. Configuration average of T_i.
2. Configuration average of KE_i (= 2×T_i − F_i^2).
3. Configuration average of F_i^2.
4. Configuration average of local part of potential energy.
5. Configuration average of short range part of MPC energy.
6. Configuration average of long range part of MPC energy.
7. Configuration average of non-local energy.
8. Configuration average of local energy.
9. Configuration average of electron-ion CPP energy.
10. Configuration average of electron CPP energy.
11. Configuration average of electron-electron CPP energy.
12. Average weight of configuration (for branching DMC).
13. Minimum weight of configuration (for branching DMC).
14. Maximum weight of configuration (for branching DMC).
15. Average age of the configurations (number of time steps).
16. Maximum age of a configuration (number of time steps).
17. Effective timestep.

All energies are quoted in Hartrees per primitive cell (per electron for electron phases, per particle for electron-hole phases). The `reblock` utility should be used to analyze the dmc.hist and dmc.hist2 files. Information about the different components of the energy and how they are calculated may be found in Sections 10.16–10.20.4.

8 Specifying the Slater determinants

8.1 Gaussian basis sets

The file gwfn.data contains the information about the orbitals in the determinant(s). The gwfn.data file contains the orbitals, which will have been generated by GAUSSIAN94/98/03 or CRYSTAL95/98/03, and other information is added by the utilities provided in the directory utils/wfn_converters.

The gwfn.data specifies a reference configuration, which may consist of one or more determinants. The *input* file allows use of either the reference configuration (GS) or to specify excitations/additions/subtractions from GS. If the excitations/additions/subtractions are made to a single determinant GS configuration then we have to specify SD in the INPUT file and if they are made to a multi-determinant GS configuration we have to specify MD.

The simplest case is when we want to use exactly the GS configuration specified in gwfn.data. The INPUT file could then contain:

```
32 32   ! NEU NED 32 spin up and down electrons
GS      ! Ground state calculation as specified in gwfn.data
```

Excitations/additions/subtractions are using the keywords 'PR', 'PL' and 'MI', respectively. Several changes to the GS configuration can be made at once. For example:

```
32 32                ! NEU NED 32 spin up and down electrons
SD                   ! Single determinant calculation
DET 1 1 PR 2 1 5 1   ! Promote (PR) electron in determinant
                     ! 1 spin 1 ("up") band 2 k-point 1, to band 5
                     ! k-point 1 in determinant 1, spin 1 ("up")
DET 1 2 PL 6 1       ! Add electron spin 2 ("down") to det 1,
                     ! band 6 k-point 1
DET 1 2 MI 6 1       ! Remove the added electron
```

Here is an example for a multi-determinant case:

```
32 32                ! NEU NED 32 spin up and down electrons
MD                   ! Multi-determinant calculation
3                    ! 3 determinants
1d0                  ! Determinant 1 prefactor
1d0                  ! Determinant 1 prefactor
1d0                  ! Determinant 1 prefactor
DET 3 1 PR 2 1 5 1   ! Promotion as examples above
```

Note that fancier orbital occupation schemes allowing finer control of which orbitals are used will be introduced shortly, along the lines of the one now used in the case of plane-wave basis sets [see next section].

8.2 Plane wave basis

The file pwfn.data specifies the plane wave orbitals used in a CASINO plane wave calculation. These orbitals are usually generated using a density functional computation and the data file contains information on the geometry, the k-points, the reciprocal lattice vectors and the expansion coefficients of Kohn-Sham orbitals for several bands, as well as the corresponding Kohn-Sham energies.

The pwfn.data file typically also contains orbitals that are not, or only partially occupied in the density functional computation. CASINO being a Monte Carlo program dealing with real electrons is based on Slater determinants of orbitals containing zero or one electron per spin. The file 'input' specifies which orbitals of pwfn.data make up the Slater determinant(s) used by CASINO.

8.2.1 The ground state

In many cases one wishes to use a single Slater determinant occupying the lowest lying Kohn-Sham orbitals. In this case the file 'input' might contain

neu : 32
ned : 32

specifying the number of electrons and

%block wavefunction
GS
%endblock wavefunction

GS is not used with any other input token and instructs CASINO to set up a single ground state determinant according to its in-build defaults (see below). The neu and ned parameters can be varied independently in units of one from 0 to the maximum number that is consistent with the supplied Kohn-Sham orbitals. For example

neu : 32
ned : 31

and
%block wavefunction
GS
%endblock wavefunction

yields a 63 electron system. This system is similar to the one above except for an electron hole (assuming that the underlying neutral system is actually a 64 electron system). Often, the GS keyword will be all you need. However, if the Kohn-Sham energy that this hole corresponds to appears at several k-points or bands the ground state occupation is not unique. We have several degenerate possibilities and CASINO chooses one using its default scheme. Note that such degeneracies can also appear for a neutral ground state.

8.2.2 Choosing between degenerate ground states

Since degeneracies are not uncommon there are several options for altering CASINO's defaults to control the population of orbitals. The token GS allows no further tokens. We have to use

%block wavefunction
SD
GSPEC IP 0.0 0 0 1
%endblock wavefunction

SD on its own is identical to GS. However, now we can use GSPEC to specify how the ground state should be populated.

GSPEC

GSPEC is followed by a two letter token (IP, P1, P2, and UP) determining the internal variable POL, a phase between zero and 2π and three flags (zero or one). IP refers to occupying the up and down electrons separately until the neu lowest up electrons are occupied and the ned lowest down electrons too. IP can be replaced by P1/P2 and UP. In these cases CASINO first evaluates the total number of electrons in the system ntot=neu+ned. It continues to occupy the lowest lying orbitals. In the case of degeneracies it aims to

P1: occupy as many up electrons as possible.

P2: occupy as many down electrons as possible.

UP: align the number of occupied electrons as closely as possible with neu and ned.

The phase specified by GSPEC is the default value used instead of the built in default. The three following flags are

DIR If set to one the orbitals will be occupied from first to last used k-point as they appear in pwfn.data, and reverse if set to zero.

SPR If set to one CASINO attempts to spread the occupied orbitals over several k-points, and over as few as possible if set to zero.

SPRBND Same as SPR except that now the focus is on different bands at the same k-point.

These flags obviously only take effect if the highest occupied energy levels are not all occupied, i.e., if there are degeneracies. In that case GSPEC can be used efficiently to generate different occupation schemes without using any promotions (see below).

8.2.3 Multi-determinant calculations

In a multiple determinant calculation, i.e., using MD instead of SD followed by the number of determinants and their relative weights

```
%block wavefunction
MD
3
0.5
0.7
0.3
GSPEC IP 0.0 0 0 1
%endblock wavefunction
```

GSPEC sets the default values for all determinants. If it is omitted CASINO uses built in defaults. DPSEC can then be used instead of GSPEC if the user wishes to set flags only for a specific determinant. The syntax is equivalent to GSPEC except for two integers in the place of the two letter token. These are ISPIN and IDET. ISPIN may be 1 or 2 referring to up and down spin determinants and IDET can take an integer value from one to the maximum set by MD, i.e

```
DSPEC 1 2 0.0 0 0 1
```

sets the phase, DIR, SPR, and SPRBND for the up-electron sub-determinant in determinant number 2. In addition, the value for POL can be specified by using the line
POL token ALL
or
POL token number
where token can be chosen from IP, P1, P2, and UP. Using ALL results in POL being set for all determinants. Using a number instead means POL will only be set for the determinant given by the number.

The occupation of orbitals can be further modified by using the the specifier EPOL:

EPOL -0.32800768 ALL

or
POL -0.32800768 number
EPOL temporarily lowers the energy eigenvalues of the down electron Kohn-Sham eigenvalues by an amount given by the real number that follows the token. In effect it biases the occupation scheme and can be used to generate strongly spin polarized systems.

If the use of EPOL (or any other flag) leads to the number of spin-up and spin-down electrons not being equal to neu and ned CASINO stops and prints out the number of electrons to which neu and ned have to be set.

8.2.4 Phases

Why phases? We know that for many, but not all orbitals we can extract two real orbitals (the real and the imaginary part of a complex orbital) that we can use to build a determinant. In some cases we might only use one of the two. We could use the real or the imaginary part, however, we could also use some linear combination of the two. It is possible to achieve this by using a multiple determinant calculation, one involving the real part and one involving the imaginary part in conjunction with an appropriate weighting of the determinants. Since one determinant is more efficient than several CASINO allows the specification of phases to choose the linear combination of real and imaginary part, see Section 10.12.

GSEPC and DSPEC modify the phases for a whole determinant. ASINGLE modifies the phase of a single orbital. ASINGLE is followed by a phase between zero and 2π, followed by a spin, a determinant, a k-point and a band index, i.e.,

ASINGLE 0.5 2 3 2 2

sets the phase to 0.5 for the orbital at k-point 2, band 2 in the 3rd determinant for the down spin electrons. This is independent of whether this orbital is actually used. The phases are specified in multiples of π. The units for the phases can be altered using UNITA followed by a real number:

UNITA 1.0

Now setting a phase to $\frac{1}{2}\pi$ as in the example above has to be down using

ASINGLE 1.570796327 2 3 2 2

Instead of setting the phases for single orbitals it is possible to set phases for an entire block using ABL:

ABL 0.5 2 3 1 1 2 2

This command has the same effect as the ASINGLE instruction above except that the the last four numbers refer to the minimum k-point and minimum band index and the maximum k-point and maximum band index of the block, in this order.

8.2.5 Promotions

GSPEC, DPSEC, POL and EPOL are very flexible, however, using a few flags is unlikely to cover all possible occupations of the given orbitals, so once the number of electrons has been set, promotions can be used to generate any possible occupation scheme. The syntax is

DET idet ispin PR bfrom kfrom bto kto

ispin and idet determining which determinant and spin are being dealt with. bfrom and kfrom refer to the band and k-point of the electron that is to be moved, bto and kto are the positions the electron is to be moved to.

Promotions my be combined with setting phases by using
DET idet ispin PRA bfrom kfrom phfrom bto kto phto
If applicable, phfrom is the phase to be used for the electron remaining at bfrom and kfrom and phto is the phase of the target orbital, e.g.
DET 1 3 PRA 4 3 0.2 4 4 0.2

8.2.6 Defaults

No values except the number of up and down electrons have to be specified. The CASINO defaults for all determinant are

POL = -1
DIR = SPR = SPRBND = 1 (i.e., true)
UNITA = π
EPOL = 0.0
All phases are set to 0.0.

8.3 Blip and spline basis

Note that the setup for the blip function module and the spline function module has not yet been fully merged with the plane wave setup routines, so that one cannot currently use the fancy orbital occupation schemes, excited states, or multi-determinant wave functions with blip or spline orbitals. This functionality will almost certainly be included shortly (or even sooner if someone specifically requests it).

9 Pseudocode

9.1 Subroutine VMC

```
Loop over blocks of time steps.
  Loop over time steps.
     Loop over configurations.
        Loop over electron in configuration.
           - Calculate local energy at R_old, if required.
           - Propose move of electron.
           - Calculate ratio Ψ(R_new)/Ψ(R_old).
           - Calculate local energy at R_new, if required.
           - Perform Metropolis accept/reject step.
           - If move is accepted update cofactor matrix and determinant.
           - Accumulate local energy etc., if required.
        End loop over electrons in configuration.
     End loop over configurations.
  End loop over time steps.
End loop over blocks of time steps.
```

9.2 Subroutine DMC

```
Loop over blocks of time steps.
  Loop over moves in a block.
     Loop over configurations.
        Loop over electrons in configuration.
           - Evaluate drift vector at old position.  Use one of various schemes
           to evaluate corresponding limited vector.
           - Evaluate diffusive component of the electron displacement.
           - Generate new position of electron (using limited drift vector).
           - If crossed node then reject move of this electron.
           - Evaluate drift vector at new position.  Use one of various schemes
           to evaluate corresponding limited vector.
           - Use limited drift velocities to calculate Metropolis acceptance probability
           for proposed electron move.  Decide whether or not move is accepted.
           - Evaluate components of limited electron energy at whichever
           position (old or new) that it ends up at.
        End loop over electrons in configuration.
        - Evaluate local energy of the new configuration and the limited energy
        - Calculate branching factor for new configuration and so determine
        the number of configurations that proceed to next generation at
        point in phase space occupied by this configuration.  Use the
        limited energy for evaluating the branching factor.
     End loop over configurations.
     - Update best estimate of energy.
     - Update reference energy.
  End loop over moves in a block.
End loop over blocks of time steps.
```

10 Theoretical background

10.1 The trial wave function

Throughout this section we assume the trial wave function is of the Slater-Jastrow form

$$\psi_T(\mathbf{R}) = \exp(J(\mathbf{R}))D_\uparrow(\mathbf{r}_1,\ldots,\mathbf{r}_{N_\uparrow})D_\downarrow(\mathbf{r}_{N_\uparrow+1},\ldots,\mathbf{r}_N), \tag{1}$$

where $\mathbf{R} = (\mathbf{r}_1,\ldots,\mathbf{r}_N)$ is a point in the configuration space of the N-electron system, J is the Jastrow function and $D_\uparrow$ and $D_\downarrow$ are Slater determinants of orbitals of spin-up and spin-down electrons respectively. See reference [23] for further information.

10.2 The variational Monte Carlo method

10.2.1 Evaluating expectation values

The expectation value of the Hamiltonian $\hat{H}$ with respect to the trial wave function ψ_T can be written as

$$\langle\hat{H}\rangle = \frac{\int E_L(\mathbf{R})\psi_T^2(\mathbf{R})\,d\mathbf{R}}{\int \psi_T^2(\mathbf{R})\,d\mathbf{R}}, \tag{2}$$

where $E_L(\mathbf{R}) = \psi_T^{-1}(\mathbf{R})\hat{H}(\mathbf{R})\psi_T(\mathbf{R})$ is the *local energy*.

We can evaluate this expectation value by using the Metropolis algorithm [18] to generate a sequence of configurations $\mathbf{R}$ distributed according to $\psi_T^2(\mathbf{R})$ and averaging the corresponding local energies.

10.2.2 The electron-by-electron algorithm

The implementation of VMC in CASINO involves repeatedly sweeping through the set of electrons, making a trial move of each in turn and accepting or rejecting the move in accordance with the Metropolis algorithm. The Metropolis transition probability density is Gaussian of variance τ, where τ is the VMC timestep (DTVMC).

In the "standard" Metropolis algorithm quantities to be averaged are evaluated at the end of each configuration move (that is, after all the electrons have attempted to move). However, quantities may be evaluated after each individual electron move so long as care is taken to ensure that, for a randomly chosen step, the probability of arriving at a particular configuration $\mathbf{R}$ having just attempted to move a particular electron i is

$$p_i(\mathbf{R}) = \psi_T^2(\mathbf{R})/N. \tag{3}$$

Calculating quantities in this way has the advantage that we can average over those trial electron moves that are rejected. Suppose for a particular electron move the probability of acceptance is p_{acc} and the energies[9] of the electron in the existing[10] and proposed configurations are ϵ_{old} and ϵ_{new} respectively. Then, irrespective of whether or not the move is accepted, we can evaluate the contribution to the average from this particular electron as

$$\epsilon = (1 - p_{\mathrm{acc}})\,\epsilon_{\mathrm{old}} + p_{\mathrm{acc}}\epsilon_{\mathrm{new}}. \tag{4}$$

Calculating quantities using Equation 4 allows the contribution of configurations with high energies but low probabilities (such as when two electrons come close together) to be more readily included in averages: fluctuations in the system energy caused by the occasional acceptance of such unlikely configurations are reduced, decreasing the error bars on the estimated quantities [19].

On the other hand, in the limit of perfect importance sampling (that is, where the trial wave function is exact and so the local energy is constant in configuration space), the "standard" Metropolis

[9] By the energy of an electron we mean the sum of its kinetic energy, electron-ion potential energy and half the sum of its electron-electron interaction energy. Summing these over all electrons we get the total energy of the configuration.

[10] Note that an electron energy in the "existing" configuration is *not* the same as the electron's energy at the end of the previous sweep: some of the other electrons will almost certainly have been moved in the intervening period. Only the electron-ion component of energy (ignoring core-polarization effects in pseudoatoms) will be the same.

algorithm gives the correct result with zero variance. Evaluating the electron energies as we sweep through the set gives a result with non-zero variance because each time an electron energy is evaluated, we will (if the electron move is accepted) change the energies that the preceding electrons *should* have in the resulting configuration.

In summary, better trial wave functions favour the standard Metropolis method.

In addition, if the electron energies are very expensive to evaluate then it may be preferable to evaluate them once per electron at the end of the configuration move rather than twice per electron, as required for use with Equation 4.

We refer to the standard Metropolis procedure as method 1 and the method in which quantities are calculated during the sweep as method 2. Both methods are available in CASINO: the VMC_METHOD input parameter selects which is to be used.

The local energy does not have to be calculated every configuration move. In particular, energies are not calculated during the first NEQUIL moves of a VMC simulation when the Metropolis algorithm has yet to reach its equilibrium. Furthermore, because the configurations are serially correlated, the expense of calculating the energy at every configuration move is unjustified: it is more efficient to calculate the energy once every CORPER moves, where typically the input parameter CORPER might be 4 or 5.

Note that it is not necessary to write out every local energy calculated. NVMCAVE successive energies can be averaged over before being written out to `vmc.hist`: this helps ensure that the `vmc.hist` file is not excessively large when accurate calculations are carried out. Furthermore, on parallel machines, the local energies computed on each processor node are averaged over before being written out.

The input parameter NMOVE gives the number of energies written out to the `vmc.hist` file in one "block" of moves. Thus, for a single processor and single block of moves, the total number of configuration moves made is in fact NMOVE×NVMCAVE×CORPER.

10.2.3 Two-level sampling

The Metropolis acceptance probability for a move from $\mathbf{R}'$ to $\mathbf{R}$ in the standard algorithm is

$$p(\mathbf{R} \leftarrow \mathbf{R}') = \min\left\{1, \frac{\psi_T^2(\mathbf{R})}{\psi_T^2(\mathbf{R}')}\right\} = \min\left\{1, \frac{D_\uparrow^2(\mathbf{R}) D_\downarrow^2(\mathbf{R}) \exp(2J(\mathbf{R}))}{D_\uparrow^2(\mathbf{R}') D_\downarrow^2(\mathbf{R}') \exp(2J(\mathbf{R}'))}\right\}. \tag{5}$$

It is straightforward to show that if detailed balance [23] in configuration space is satisfied then the resulting ensemble of configurations will be distributed according to the square of the trial wave function.

However, `CASINO` employs a two-level sampling algorithm, which has been shown to be considerably more efficient [21].

Let us define the first-level acceptance probability

$$p_1(\mathbf{R} \leftarrow \mathbf{R}') = \min\left\{1, \frac{D_\uparrow^2(\mathbf{R}) D_\downarrow^2(\mathbf{R})}{D_\uparrow^2(\mathbf{R}') D_\downarrow^2(\mathbf{R}')}\right\} \tag{6}$$

and the second-level acceptance probability

$$p_2(\mathbf{R} \leftarrow \mathbf{R}') = \min\left\{1, \frac{\exp(2J(\mathbf{R}))}{\exp(2J(\mathbf{R}'))}\right\}. \tag{7}$$

In the two-level algorithm we accept trial moves from $\mathbf{R}'$ to $\mathbf{R}$ with probability $p_1(\mathbf{R} \leftarrow \mathbf{R}')p_2(\mathbf{R} \leftarrow \mathbf{R}')$. It can be shown that, provided detailed balance in configuration space is satisfied, this procedure also results in an ensemble of configurations distributed according to the square of the trial wave function.

The two-level approach is computationally advantageous because the Metropolis accept/reject step can be carried out in two stages: if the "first level" is accepted (with probability p_1) then we compute the Jastrow functions of the new and old configurations and determine whether the "second level" is

accepted (with probability p_2). Thus, if a move is rejected at the first level then we do not have to compute the Jastrow functions for the new and old configuration[11].

It is unclear how two-level sampling should be applied in the context of VMC method 2 since we do not know the full acceptance probability for the trial electron move if the move is rejected at the first level. Therefore, when VMC_METHOD is 2, single-level sampling is used for those moves on which electron energies are calculated (with $p_1 p_2$ being the acceptance probability); two-level sampling is used elsewhere.

10.2.4 The optimal value of the VMC timestep

The VMC timestep DTVMC should be chosen such that the correlation period (as determined by reblocking analysis) of the resulting configuration local energies is minimized. For large timesteps the move rejection probability is high which obviously leads to serial correlation. On the other hand, for low timesteps the configuration does not move very far at a given step, and so serial correlation is again large.

A rule of thumb for choosing an appropriate DTVMC is that the average acceptance probability should be about 1/2. The average acceptance probability converges very rapidly to its final value; hence by making a couple of extremely short trial VMC runs, it is easy to find an appropriate value for DTVMC. If you use at least 500 steps in the VMC equilibration phase, then CASINO will automatically optimize the VMC time step if you set the keyword OPT_DTVMC to 1.

10.2.5 Automatic DTVMC optimization

Optimization of DTVMC to achieve an acceptance ratio of (roughly) 50% is performed in CASINO by means of linear inter/extrapolation of two consecutive trial pairs of values $(\log(DTVMC), Acc.Ratio)$. As we are dealing with random electronic configurations, these numbers are subjected to fluctuations, due to which divergences (i.e., vertical lines arising from the interpolation) are bound to appear near the solution. In order to avoid these, for any two consecutive DTVMC values that are too close from each other, the next trial point is chosen depending only on the last one. Also, for acceptance ratios too far away from 50% the function becomes irregular, so a similar method is used.

There are some remarks that must be made regarding the optimum value of DTVMC, the most important one being that the appropriate way to determine it is by minimizing the correlation time of the energy of the electronic configurations. Nevertheless, it is found in practice that taking the acceptance ratio to 50% yields almost the same results, but with the advantage of being much faster to compute, because the energy is not calculated, and can thus be implemented into the equilibration process.

A second remark is that DTVMC can be regarded as a length (it measures the width of the probability distribution that is sampled to move the particles at every step), and as such, is connected with a characteristic length of the system, which can be influenced by its size, density, etc. For example, for low densities, it is found that the optimized DTVMC is much larger than those that one gets used to seeing; what happens is that the distribution flattens into a constant (or almost so) so that the electrons can move to a random place in the simulation box (not only within a small environment of their previous positions), hence better significance (decorrelation) in the sampling of the desired observables is achieved.

The last comment is that the procedure of optimization has not been proven infallible for all initial values or extreme conditions, such as systems in which the small number of particles might prevent gathering of good statistics for the interpolation. Nor is it the case that there is an analytic connection of the acceptance ratio being 50% with the minimization of the correlation time. However, test cases show that the technique works and is useful, specially when tackling a new system for the first time.

10.3 The diffusion Monte Carlo method

See, e.g., references [22] and [23] for general information about DMC. Here we present some more detailed information about the implementation of DMC within CASINO. Our algorithm follows that of

[11] It is better to use the Slater wave function to compute the first-level acceptance probability because p_1 is much lower (in general) than the corresponding acceptance probability for the Jastrow part (p_2); hence we are less likely to need to compute the second level acceptance probability than if the situation were reversed.

Umrigar, Nightingale, and Runge [5] (referred to simply as UNR), with some additional developments due to Umrigar and Filippi [7].

10.3.1 Imaginary-time propagation

Let $\mathbf{R}$ be a point in the configuration space of an N-electron system. The importance-sampled DMC method propagates the distribution $f(\mathbf{R},t) = \Psi(\mathbf{R})\Phi(\mathbf{R},t)$ in imaginary time t, where Ψ is the trial wave function and Φ is the DMC wave function. The importance-sampled imaginary-time Schrödinger equation may be written in integral form,

$$f(\mathbf{R},t) = \int G(\mathbf{R} \leftarrow \mathbf{R}', t-t') f(\mathbf{R}',t')\, d\mathbf{R}' \,, \tag{8}$$

where the Green's function $G(\mathbf{R} \leftarrow \mathbf{R}', t-t')$ satisfies the initial condition

$$G(\mathbf{R} \leftarrow \mathbf{R}', 0) = \delta(\mathbf{R}-\mathbf{R}') \,. \tag{9}$$

The Green's function used in DMC is an approximation to the exact form which is accurate for short time steps, $\tau = t - t'$ (DTDMC),

$$G_{\mathrm{DMC}}(\mathbf{R} \leftarrow \mathbf{R}', \tau) = G_{\mathrm{D}}(\mathbf{R} \leftarrow \mathbf{R}', \tau)\, G_{\mathrm{B}}(\mathbf{R} \leftarrow \mathbf{R}', \tau) \,, \tag{10}$$

where

$$G_{\mathrm{D}}(\mathbf{R} \leftarrow \mathbf{R}', \tau) = \frac{1}{(2\pi\tau)^{3N/2}} \exp\left(-\frac{(\mathbf{R}-\mathbf{R}'-\tau\mathbf{V}(\mathbf{R}'))^2}{2\tau} \right) \tag{11}$$

is the drift-diffusion Green's function and

$$G_{\mathrm{B}}(\mathbf{R} \leftarrow \mathbf{R}', \tau) = \exp\left(-\frac{\tau}{2}[E_L(\mathbf{R}) + E_L(\mathbf{R}') - 2E_T] \right) \tag{12}$$

is the branching Green's function. E_T is the reference energy, which acts as a renormalization factor, see Section 10.5. E_L is the local energy,

$$E_L(\mathbf{R}) = \Psi^{-1}\hat{H}\Psi, \tag{13}$$

where $\hat{H}$ is the Hamiltonian, and $\mathbf{V}$ is the drift vector,

$$\mathbf{V}(\mathbf{R}) = \Psi^{-1}\nabla\Psi. \tag{14}$$

Note that $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_N)$, where $\mathbf{v}_i$ is the drift vector of electron i.

10.3.2 The ensemble of configurations

The f distribution is represented by an ensemble of electron configurations which are propagated according to rules derived from the Green's function of Eq. 10. G_{D} represents a drift-diffusion process while G_{B} represents a branching process. The branching process leads to fluctuations in the population of configurations and/or fluctuations in their weights.

We will introduce labels for the different configurations α present at each time step m. From now on $\mathbf{R}$ represents the electronic positions of a particular configuration in the ensemble, and i labels a particular electron.

In CASINO electrons are moved one at a time. This is much more efficient than making whole configuration moves for large systems. This is standard procedure for large systems, but most of the algorithms described in the literature are for moving all electrons at once.

10.4 Drift and diffusion

We now discuss the practical implementation of the drift-diffusion process using the *electron-by-electron* algorithm in which electrons are moved one at a time, as used in `CASINO`.

To implement the drift-diffusion step, each electron i in each configuration α is moved from $\mathbf{r}'_i(\alpha)$ to $\mathbf{r}_i(\alpha)$ in turn according to

$$\mathbf{r}_i = \mathbf{r}'_i + \chi + \tau \mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N), \tag{15}$$

where χ is a three-dimensional vector of normally distributed numbers with variance τ and zero mean. $\mathbf{v}_i(\mathbf{R})$ denotes those components of the total drift vector $\mathbf{V}(\mathbf{R})$ due to electron i.

Hence each electron i is moved from $\mathbf{r}'_i$ to $\mathbf{r}_i$ with a transition probability density of

$$t_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N) = \frac{1}{(2\pi\tau)^{3/2}} \exp\left(\frac{(\mathbf{r}_i - \mathbf{r}'_i - \tau \mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N))^2}{2\tau} \right). \tag{16}$$

For a complete sweep through the set of electrons, the transition probability density for a move from $\mathbf{R}' = (\mathbf{r}'_1, \ldots, \mathbf{r}'_N)$ to $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ is simply the probability that each electron i moves from $\mathbf{r}'_i$ to $\mathbf{r}_i$. So the transition probability density for the configuration move is

$$T(\mathbf{R} \leftarrow \mathbf{R}') = \prod_{i=1}^{N} t_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N). \tag{17}$$

In the limit of small timesteps, the drift velocity $\mathbf{V}$ is constant over the (small) configuration move. Evaluating the product in this case, we find that the transition probability density is

$$T(\mathbf{R} \leftarrow \mathbf{R}') = G_D(\mathbf{R} \leftarrow \mathbf{R}', \tau), \tag{18}$$

so that the drift-diffusion process is described by the drift-diffusion Green's function G_D.

At finite timesteps, however, the approximation that the drift velocity is constant leads to the violation of the detailed balance condition.

We may enforce the detailed balance condition on the DMC Green's function by means of a Metropolis-style accept/reject step introduced by Ceperley *et al* [33]. This has been shown to greatly reduce timestep errors [4]. The move of the ith electron of a configuration is accepted with probability

$$\begin{aligned}
& \min\left\{1, \frac{t_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i \leftarrow \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)}{t_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N)}\right\} \\
= \quad & \min\left\{1, \exp\left[\left(\mathbf{r}'_i - \mathbf{r}_i + \frac{\tau}{2}(\mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N) - \mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)\right)\right.\right. \\
& \qquad \left.\left. \cdot \left(\mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N) + \mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)\right)\right] \right. \\
& \qquad \left. \times \frac{\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)}{\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N)}\right\} \\
\equiv \quad & a_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N).
\end{aligned} \tag{19}$$

This leads to the single-electron detailed balance condition

$$\begin{aligned}
& s_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N) \\
= \quad & s_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i \leftarrow \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N),
\end{aligned} \tag{20}$$

where

$$\begin{aligned}
s_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N) \quad = \quad & a_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N) \\
& \times t_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N),
\end{aligned} \tag{21}$$

is the effective single-electron transition probability density, once the accept/reject step has been introduced.

Hence we find that the effective transition probability density for the entire configuration move satisfies

$$
\begin{aligned}
S(\mathbf{R} \leftarrow \mathbf{R}') &= \prod_{i=1}^{N} s_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i \leftarrow \mathbf{r}'_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N) \\
&= \prod_{i=1}^{N} s_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i \leftarrow \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N) \frac{\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_i, \mathbf{r}'_{i+1}, \ldots, \mathbf{r}'_N)}{\psi_T^2(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_N)} \\
&= S(\mathbf{R}' \leftarrow \mathbf{R}) \frac{\psi_T^2(\mathbf{R})}{\psi_T^2(\mathbf{R}')} \qquad (22)
\end{aligned}
$$

And so detailed balance in configuration space is satisfied.

It is more efficient to use an electron-by-electron algorithm than the (perhaps more straightforward) *configuration-by-configuration* algorithm in which moves of entire configurations are proposed and then accepted or rejected. This is because, for a given timestep, a configuration will travel further on average if the accept/reject step is carried out for each electron in turn. For example, it is clear that it is very unlikely for a configuration not to be moved at all in an electron-by-electron algorithm. Hence the sampling of configuration space in an electron-by-electron algorithm is more efficient.

After each move of each electron we check whether the configuration has crossed the nodal surface (by checking the sign of the Slater part of the trial wave function). If it has then the move is rejected. This has been found to be the least-biased method of imposing the fixed-node approximation [23].

10.5 Branching and population control

The branching Green's function can be implemented by altering the population of configurations and/or their weights. At the start of the calculation one chooses a target population, M_0, and the actual population $M_{\rm tot}(m)$ (see Eq. 27) is controlled so that it does not deviate too much from M_0. The population control is principally exerted by altering the reference energy, $E_T(m)$. Large changes in E_T can lead to a bias and therefore it is varied smoothly over the simulation.

For each move of all the electrons in configuration α the branching factor is calculated as:

$$
M_b(\alpha, m) = \exp\left[\left(-\frac{1}{2}\left\{S(\mathbf{R}_{\alpha,m}) + S(\mathbf{R}'_{\alpha,m})\right\} + E_T(m)\right)\tau_{\rm eff}(\alpha, m)\right] \qquad (23)
$$

where $\tau_{\rm eff}$ is the effective time step for configuration α at time step m (see Section 10.6), S is the local energy (we denote it by S because it is usually a modified version of E_L, see Section 10.6). Unless weighted DMC is used (i.e., unless LWDMC=.TRUE.), the number of copies of this configuration that continue to the next time step is given by:

$$
M(\alpha, m) = \mathrm{INT}\{\eta + M_b(\alpha, m)\}, \qquad (24)
$$

where η is a random number drawn from a uniform distribution on the interval [0,1].

In weighted DMC, each configuration carries a weight that is simply multiplied by $M_b(\alpha, m)$ after each move; only if the weight of a configuration goes outside certain bounds (currently above 2 or below 0.5) is it allowed to branch or be combined with another configuration.

Throughout this section we denote the best estimate of the ground-state energy at time step α by $E_{\rm best}(m)$. At the start of a DMC run we set $E_{\rm best}(0) = E_T(0) = E_{\rm V}$, where $E_{\rm V}$ is the average local energy of the initial configurations. During equilibration $E_{\rm best}$ is updated after each *block* of moves as:

$$
E_{\rm best}(m) = \frac{3}{4}E_{\rm block} + \frac{1}{4}E_{\rm best}(m-1). \qquad (25)
$$

At the end of each block various book-keeping tasks are performed, i.e., in VMC data is written to file at the end of each block and in DMC data is written out, the population may be renormalized (POPRENORM) and $E_{\rm best}$ is updated during equilibration. E_T is updated after every time step as

$$
E_T(m+1) = E_{\rm best}(m) - \frac{g^{-1}}{\tau_{\rm EFF}(m)} \log\left(\frac{M_{\rm tot}(m)}{M_0}\right), \qquad (26)
$$

where $g^{-1} = \min\{1, \tau c_{E_T}\}$, c_{E_T} is a constant which must be set in the INPUT file (CEREFDMC), but is usually set equal to one, $M_0 = \text{NCONFIG} \times \text{NNODES}$ is the target number of configurations, and

$$M_{\text{tot}}(m) = \sum_{\alpha=1}^{N_{\text{config}}(m)} w_\alpha(m), \tag{27}$$

where $N_{\text{config}}(m)$ is the number of configurations and w_α is the weight of configuration α. Note that $E_{\text{best}}(m)$ is the best energy at time step m while $E_T(m+1)$ is the trial energy to be used in the next time step. τ_{EFF} is the current best estimate over all configurations and time steps of the mean effective time step, calculated using Eq. 54 with $\hat{A} = \tau_{\text{eff}}(\alpha, m)$. Note that g is the time scale (in terms of time steps) over which the population attempts to return to M_0. During accumulation E_{best} is set equal to the current value of the mixed estimator of the energy, given by Eq. 54, with $\hat{A} = E_L(\alpha, m)$.

To further limit the size of population fluctuations, the population can be renormalized to M_0 at the start of each block (POPRENORM). This should not be necessary, however, and should be avoided if at all possible as it can lead to large time step errors.

10.6 Modifications to the Green's Function

10.6.1 The effective time step

Time-step errors can be reduced and the stability of the DMC algorithm improved by modifying the Green's function. An important modification is to introduce an effective time step, τ_{eff}, into the branching factor [4]. When the accept/reject step is included, the mean distance diffused by each electron each move (which should go as the square root of the time step) is reduced because some moves are rejected. When calculating branching factors it is therefore more accurate to use a time step appropriate for the actual distance diffused. Umrigar and Filippi [7] have suggested using an effective time step for each configuration at each time step. This helps to eliminate the problem of "persistent" configurations, which are low energy configurations for which all moves are rejected, and which can multiply and bias the calculation. The effective time step is given by

$$\tau_{\text{eff}}(\alpha, m) = \tau \frac{\sum_i p_i \Delta r_{\text{d},i}^2}{\sum_i \Delta r_{\text{d},i}^2}\ , \tag{28}$$

where the averages are over all attempted moves of the electrons i in configuration α at time step m. The $\Delta r_{\text{d},i}$ are the diffusive displacements (i.e., the distance travelled by the electrons *without* the drift-displacement, see Eq. 15) and p_i is the acceptance probability of the electron move, see Eq. 19. The values averaged over the current run are written in the output file.

We calculate $\tau_{\text{EFF}}(m)$ using Eq. 54 with $\hat{A} \equiv \tau_{\text{eff}}(\alpha, m)$.

10.6.2 Drift vector and local energy limiting

The drift vector diverges at the nodal surface and a configuration which approaches a node can exhibit a very large drift, resulting in an excessively large move in the configuration space. One can improve the Green's function by cutting off the drift vector when its magnitude becomes large. The total drift vector is defined in Eq. 14.

We use the smoothly cut-off drift vector suggested by UNR [5]. For each electron with drift vector $\mathbf{v}_i$, we define the smoothly cut-off drift vector $\tilde{\mathbf{v}}_i$, where:

$$\tilde{\mathbf{v}}_i = \frac{-1 + \sqrt{1 + 2a|\mathbf{v}_i|^2\tau}}{a|\mathbf{v}_i|^2\tau}\mathbf{v}_i\ , \tag{29}$$

where a is a constant which can be chosen to minimize the bias (in UNR [5] a value of $a = 1/4$ was suggested for all-electron calculations, but $a = 1$ may be more appropriate for pseudopotential calculations). The value of $a =$ ALIMIT must be entered by the user if NUCLEUS_GF_MODS is set to `false`; otherwise a will be calculated as described in Section 10.7.

In the UNR [5] scheme the modified local energy, $S(\alpha, m)$, is given by

$$S(\alpha, m) = E_{\text{best}}(m) - [E_{\text{best}}(m) - E_L(\alpha, m)]\frac{|\tilde{\mathbf{V}}|}{|\mathbf{V}|}\ , \tag{30}$$

where $\tilde{\mathbf{V}}(\alpha, m) = (\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_N)$. Note that we define S slightly differently from Ref. [5]. S is used only in the branching factor and when evaluating the average energy we sum the unlimited local energies, E_L.

Finally, a very simple option is simply to cut-off the local energy drift vector when their magnitudes becomes large using the method of Depasquale *et al.* [8],

$$\begin{aligned} S(\mathbf{R}) &= E_{\rm V} + \text{sign}[2/\sqrt{\tau}, E_L(\mathbf{R}) - E_{\rm V}] \quad \text{for } |E_L(\mathbf{R}) - E_{\rm V}| > 2/\sqrt{\tau} \\ \tilde{\mathbf{v}}_i(\mathbf{R}) &= \text{sign}[1/\tau, \mathbf{v}_i] \qquad\qquad\qquad\qquad \text{for } |\mathbf{v}_i| > 1/\tau \ . \end{aligned} \tag{31}$$

In the paper of Depasquale *et al.* [8] they recommend using the variational energy for $E_{\rm V}$, but we use the best estimate of the energy. We prefer the UNR scheme.

10.6.3 Theoretical background to the UNR limiting scheme for the drift velocity

Recall the approximation that the drift velocity remains constant over a configuration move. Unfortunately, the drift velocity varies rapidly as it diverges near the nodes. UNR suggest, therefore, that the Green's function for the drift process should be calculated on the assumption that $\nabla\psi_T$, rather than $\mathbf{V} = \psi_T^{-1}\nabla\psi_T$, is constant over a configuration move when close to the nodal surface [5].

Consider the drift of a single electron in a single configuration in the electron-by-electron DMC algorithm. Let the electron have position vector $\mathbf{r}_i(t)$ as it drifts, with all other electrons fixed at positions $\{\mathbf{r}_j\}_{j\neq i}$. Consider the case where the configuration is close to the nodal surface and $\nabla_i\psi_T(\mathbf{R})$ is nonzero at the surface[12]. Then $\mathbf{r}_i(t)$ must be close to a point $\mathbf{s}_i$ such that $\psi_T(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{s}_i, \mathbf{r}_{i+1}, \ldots, \mathbf{r}_N) = 0$. So we can write

$$\begin{aligned} \psi_T(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i(t), \mathbf{r}_{i+1}, \ldots, \mathbf{r}_N) &\approx \nabla_i\psi_T(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{s}_i, \mathbf{r}_{i+1}, \ldots, \mathbf{r}_N) \cdot (\mathbf{r}_i(t) - \mathbf{s}_i) \\ &= A r_i^{\perp}(t), \end{aligned} \tag{32}$$

where $A = |\nabla_i\psi_T(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{s}_i, \mathbf{r}_{i+1}, \ldots, \mathbf{r}_N)|$ is a constant over the move and $\mathbf{r}_i^{\perp}(t)$ is the component of $\mathbf{r}_i(t) - \mathbf{s}_i$ in the direction of $\nabla_i\psi_T(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{s}_i, \mathbf{r}_{i+1}, \ldots, \mathbf{r}_N)$.

The equation of motion for the single-electron drift process is

$$\begin{aligned} \frac{d\mathbf{r}_i(t)}{dt} &= \mathbf{v}_i(\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}_i(t), \mathbf{r}_{i+1}, \ldots, \mathbf{r}_N) = \mathbf{v}_i(t) \\ &= \frac{\nabla_i\psi_T}{\psi_T} \approx \frac{\mathbf{r}_i^{\perp}}{(r_i^{\perp}(t))^2}. \end{aligned} \tag{33}$$

Hence

$$v_i(t) = \frac{dr_i^{\perp}(t)}{dt} = \frac{1}{r_i^{\perp}(t)}. \tag{34}$$

Integrating this over one timestep, from time t to $t + \tau$, we find that

$$r_i^{\perp}(t+\tau) - r_i^{\perp}(t) \approx \sqrt{(r_i^{\perp}(t))^2 + 2\tau} - r_i^{\perp}(t) \equiv \bar{v_i}(t)\tau, \tag{35}$$

where the magnitude of the *limited* drift velocity is given by

$$\begin{aligned} \bar{v_i}(t) &= \frac{\sqrt{(r_i^{\perp}(t))^2 + 2\tau} - r_i^{\perp}(t)}{\tau} \\ &= \frac{-1 + \sqrt{1 + 2\tau/(r_i^{\perp}(t))^2}}{\tau/r_i^{\perp}(t)} = \frac{-1 + \sqrt{1 + 2v_i^2(t)\tau}}{v_i(t)\tau}. \end{aligned} \tag{36}$$

So, finally, the limited drift velocity is written as

[12] If $\nabla_i\psi_T(\mathbf{R}) = 0$ at the nodal surface then the drift velocity of electron i will be constant close to the nodal surface, rather than divergent.

$$\bar{\mathbf{v}}_i(t) = \frac{-1+\sqrt{1+2av_i^2(t)\tau}}{av_i^2(t)\tau}\mathbf{v}_i(t), \tag{37}$$

where the parameter a has been introduced such that $a = 1$ corresponds to the solution close to the node and the limit $a \to 0$ corresponds to the "normal" solution, with $\bar{\mathbf{v}}_i = \mathbf{v}_i$. a can be made position-dependent in a fashion that (roughly) distinguishes between the drift velocity being large due to proximity to a node and proximity to a nucleus. This is described in Section 10.7. However, because the dependence on a is very weak, we can simply choose a constant value of $a \in (0, 1]$ (see Section 10.6.2) if bare nuclei are not present.

Note that the limited velocity is only substantially different from the unlimited velocity when the magnitude of the latter is large (compared to $1/\sqrt{2a\tau}$). Also, the magnitude of the limited velocity is always less than the unlimited velocity. So Equation 37 is indeed a limiting scheme for the single-electron drift velocity.

By proposing electron moves using the limited drift velocity, we may improve upon the short-time approximation, since the variation of the drift velocity over moves close to nodes is taken into account. Furthermore, although the unlimited velocity diverges as the node is approached, the limited velocity remains bounded; it tends to $\sqrt{2/a\tau}$. Thus the various pathologies caused by the divergent drift velocity are ameliorated.

10.6.4 Theoretical background to the UNR limiting scheme for the local energy

Although CASINO uses an electron-by-electron DMC algorithm, the derivation below is given in the context of an entire configuration move. When close to the nodal surface, the limited energy is calculated as the average of the local energy over a *configuration* drift starting from the configuration's final position. This gives a limited local energy that remains bounded as the node is approached and is physically sensible for computing the branching factor, albeit approximate.

Consider the average value of the local energy over a configuration drift to $\mathbf{R}(t+\tau)$ from $\mathbf{R}(t) \equiv \mathbf{R}$ close to a node:

$$\begin{aligned} \bar{E_L}(\mathbf{R}) &= \frac{1}{\tau}\int_t^{t+\tau} E_L(\mathbf{R}(t'))\,dt' \\ &= E_{\text{best}} + \frac{1}{\tau}\int_t^{t+\tau} \left[E_L(\mathbf{R}(t')) - E_{\text{best}}\right]\,dt' \end{aligned} \tag{38}$$

where E_{best} is the current best DMC estimate of the ground state energy.

The divergence in the local energy close to a node can be written as

$$E_L(\mathbf{R}) = \psi_T^{-1}(\mathbf{R})\hat{H}\psi_T(\mathbf{R}) = E_{\text{best}} + \frac{B}{R_\perp} = E_{\text{best}} + BV(\mathbf{R}), \tag{39}$$

where B is a constant and $R_\perp$ is the distance of point $\mathbf{R}$ from the nearby nodal surface. Note that by the same arguments as in the single electron case, the full drift velocity is approximately given by $V(\mathbf{R}) = 1/R_\perp$.

So we find that

$$\begin{aligned} \bar{E_L}(\mathbf{R}) &= E_{\text{best}} + \frac{B}{\tau}\int_t^{t+\tau} V(\mathbf{R}(t'))\,dt' \\ &= E_{\text{best}} + \frac{B}{\tau}(R_\perp(t+\tau) - R_\perp(t)) \\ &= E_{\text{best}} + B\bar{V}(\mathbf{R}) \\ &= E_{\text{best}} + [E_L(\mathbf{R}) - E_{\text{best}}]\frac{\bar{V}(\mathbf{R})}{V(\mathbf{R})}. \end{aligned} \tag{40}$$

$\bar{E_L}$ reduces to E_L when the drift velocity is small, away from nodes and nuclei. In addition, $\bar{E_L}$ is always closer to E_{best} than E_L. Hence Equation 40 is a formula for limiting the local energy.

Since both the drift velocity and the local energy diverge as $1/R_\perp$, the limited local energy $\bar{E}_L$ remains finite as the node is approached.

In practice, even though the local energy is calculated at the end of the configuration move, the limiting scheme of Section 10.6.3 is applied to the individual single electron drift velocities. The ratio of drift velocities is therefore calculated as

$$\frac{\bar{V}(\mathbf{R})}{V(\mathbf{R})} = \frac{\left(\bar{v}_1^2(\mathbf{R}) + \ldots + \bar{v}_N^2(\mathbf{R})\right)^{1/2}}{V(\mathbf{R})}. \tag{41}$$

The limited single electron drift velocities are calculated at the same time as the local energy of the configuration, after all of the electron positions in the configuration have been updated.

10.7 Modifications to the DMC Green's function at bare nuclei

10.7.1 Modifications to the limiting of the drift velocity

The limiting of the drift velocity is intended to remove the divergence at the nodal surface; interference with the cusps at bare nuclei is an undesirable side-effect, which may introduce bias. In order to distinguish between nodes and nuclei, UNR make the a-parameter in their limiting scheme dependent on electron position. Immediately before the limited drift velocity of an electron at $\mathbf{r}'$ is calculated, a is evaluated as

$$a(\mathbf{r}') = \frac{1}{2}\left(1 + \hat{\mathbf{v}} \cdot \mathbf{e}_z\right) + \frac{Z^2 z^2}{10(4 + Z^2 z^2)}, \tag{42}$$

where $\hat{\mathbf{v}}$ is the unit vector in the direction of the unlimited drift velocity, $\mathbf{e}_z$ is the unit vector from the closest bare nucleus to the electron, z is the distance of the electron from the nucleus, and Z is the atomic number. This formula makes a small (and hence the limiting weak) if the electron is both close to the nearest nucleus and drifting towards it.

10.7.2 Preventing electrons from overshooting nuclei

In its immediate vicinity, the single-electron drift velocity is always directed towards a bare nucleus. Therefore, drifting particles should never cross the nucleus; rather, they should end up on top of it.

In order to impose this condition at finite timesteps, we work in cylindrical polar coordinates with the z-axis lying along the line from the nucleus to the electron. Let the position of the closest nucleus be $\mathbf{R}_Z$.

The position of the electron relative to the nucleus is

$$\mathbf{r}' - \mathbf{R}_Z = z'\mathbf{e}_z, \tag{43}$$

while the limited drift velocity can be resolved as

$$\bar{\mathbf{v}} = \bar{v}_z\mathbf{e}_z + \bar{v}_\rho\mathbf{e}_\rho, \tag{44}$$

where $\mathbf{e}_\rho$ is a unit vector orthogonal to $\mathbf{e}_z$.

The new z-coordinate after drifting for one timestep is

$$z'' = \max\{z' + \bar{v}_z\tau, 0\}, \tag{45}$$

which cannot lie beyond the nucleus.

The drift in the radial direction over one timestep is

$$\rho'' = \frac{2\bar{v}_\rho\tau z''}{z' + z''}. \tag{46}$$

The new radial coordinate is approximately $\bar{v}_\rho\tau$ when far from the nucleus, but it is forced to go to zero as the nucleus is approached. Hence, if the electron attempts to overshoot the nucleus, it will end up on top of it. So timestep errors caused by drifting across nuclei are eliminated.

Let the electron position at the end of the drift process be $\mathbf{r}'' = z''\mathbf{e}_z + \rho''\mathbf{e}_\rho$.

10.7.3 Diffusion close to a bare nucleus

Close to a nucleus, f is proportional to the square of the hydrogenic 1s orbital (assuming the trial wavefunction has the correct behaviour). This cusp cannot be reproduced by Gaussian diffusion at finite timesteps. In fact, starting from the nucleus, we would like our electron to take a random step $\mathbf{w}$ distributed according to $\exp(-2Z|\mathbf{w}|)$.

However, we only want to diffuse in this fashion when the electron is likely to cross the nucleus. Let Π be the plane with normal $\mathbf{e}_z$ that contains the nucleus. For the usual Gaussian diffusion process, the probability that an electron drifts (assuming that nuclear overshoot is permitted) and diffuses across Π, is

$$\tilde{q} = 1 - \tilde{p} = \frac{1}{2} \operatorname{erfc}\left(\frac{z + \bar{v}_z \tau}{\sqrt{2\tau}}\right). \tag{47}$$

So, with probability $\tilde{p}$, we sample $\mathbf{w}$ from

$$g_1(\mathbf{w}) = (2\pi\tau)^{-3/2} \exp\left(-\frac{|\mathbf{w}|^2}{2\tau}\right), \tag{48}$$

and set the new electron position to be $\mathbf{r} = \mathbf{r}'' + \mathbf{w}$; otherwise, we sample[13] $\mathbf{w}$ from

$$g_2(\mathbf{w}) = \frac{\zeta^3}{\pi} \exp(-2\zeta|\mathbf{w}|), \tag{49}$$

and set $\mathbf{r} = \mathbf{R}_Z + \mathbf{w}$. We have defined ζ by

$$\zeta = \sqrt{Z^2 + \frac{1}{\tau}}, \tag{50}$$

which reduces to Z for large timesteps, giving the desired cusp; however, this choice of ζ causes the second moments of g_1 and g_2 to be equal to $O(\tau)$. Hence the Green's function remains correct to $O(\tau)$.

The single-electron Green's function for the move from $\mathbf{r}'$ to $\mathbf{r}$ is given by

$$g(\mathbf{r} \leftarrow \mathbf{r}') = \tilde{p} g_1(\mathbf{r} - \mathbf{r}'') + \tilde{q} g_2(\mathbf{r} - \mathbf{R}_Z). \tag{51}$$

In order to calculate the Green's function for the reverse move, need to perform all of the steps above (apart from the random diffusion), starting at point $\mathbf{r}$ and ending up at $\mathbf{r}'$.

10.7.4 Using the modifications in CASINO

These three modifications to the DMC Green's function are applied if the NUCLEUS_GF_MODS keyword is set to "T". Note that they can only be used if bare nuclei are actually present!

10.8 Evaluating expectation values of observables

The *reference energy* E_T is varied to maintain a reasonably steady population. However, this procedure can result in a bias in the estimate of expectation values, especially for small populations. We evaluate expectation values using the method of UNR [5] which attempts to eliminate bias due to population control.

Using the label m for time step, Eq. 10 becomes

$$f(\mathbf{R}, m) = \int G_{\mathrm{DMC}}(\mathbf{R} \leftarrow \mathbf{R}', \tau) f(\mathbf{R}', m-1)\, d\mathbf{R}'\,. \tag{52}$$

Clearly, in the absence of the accept/reject step, the effect of including the (time-step-dependent) reference energy $E_T(m)$ in G_{DMC} can be "undone" by multiplying the right-hand-side of Eq. 52 by $\exp[-\tau E_T(m)]$. In a similar fashion the effect of including the reference energy from the previous time step can be eliminated by multiplying by $\exp[-\tau E_T(m-1)]$. When the accept/reject step is

[13] In order to sample $\mathbf{w}$ from $g_2(\mathbf{w})$, we sample the polar angle uniformly on $[0, \pi]$, the azimuthal angle uniformly on $[0, 2\pi]$ and the magnitude w from $4\zeta^3 w^2 \exp(-2\zeta w)$. This is achieved by sampling r_1, r_2 and r_3 uniformly on $[0, 1]$ and setting $w = -\log(r_1 r_2 r_3)/2\zeta$; see reference [6] for further information.

present, we can approximately undo the effect of the reference energy by using our best estimate of the effective time step $\tau_{\rm EFF}$ (See Section 10.6) in the "undoing" factors.

Continuing this process, we may eliminate the effect of changing the reference energy from $f(m)$ by multiplying it by

$$\Pi(m) = \prod_{m'=0} \exp\left[-\tau_{\rm EFF} E_T(m-m')\right] \ , \tag{53}$$

where in principle the product runs over all previous time steps. In practice it is sufficient to include T_p (=TPDMC) terms in the product, provided that T_p is greater than the number of iterations over which the DMC data are correlated by fluctuations in the reference energy: $T_p = {\rm NINT}(10/\tau)$ is generally sufficient. Let $\Pi(m,T_p) = \prod_{m'=0}^{T_p} \exp[-\tau_{\rm EFF}(m) E_T(m-m')]$. Then the mixed estimator of the expectation value of a (local) operator $\hat{A}$ may be written as:

$$\begin{aligned} \frac{\langle\Psi|\hat{A}|\Phi\rangle}{\langle\Psi|\Phi\rangle} &= \frac{\int \Psi(\mathbf{R})\hat{A}(\mathbf{R})\Phi(\mathbf{R})\,d\mathbf{R}}{\int \Psi(\mathbf{R})\Phi(\mathbf{R})\,d\mathbf{R}} \\ &\approx \frac{\sum_{m'=1}^{m}\Pi(m',T_p)\sum_{\alpha=1}^{N_{\rm config}(m')} w_\alpha(m')\hat{A}(\alpha,m')}{\sum_{m'=1}^{m}\Pi(m',T_p)\sum_{\alpha=1}^{N_{\rm config}(m')} w_\alpha(m')}, \end{aligned} \tag{54}$$

where $w_\alpha(m')$ is the weight of configuration α at the end of time step m'. (For unweighted DMC, w_α is simply the branching factor.) Note that if we choose $\hat{A}(\alpha,m) = \Psi^{-1}(\mathbf{R}_{\alpha,m})\hat{H}\Psi(\mathbf{R}_{\alpha,m})$ then Eq. 54 gives us our mixed estimator of the ground state energy at time step m. This is used as our "best estimate" of the ground state energy, $E_{\rm best}$ (see Section 10.5).

Note that the terms in the Π weights are exponential functions of the reference energy; hence the Π weights are potentially very large (or small). However, it can be seen that any constant contribution to the reference energy will cancel in Eq. 54. Therefore, in practice, we evaluate the Π-weights as:

$$\Pi(m,T_p) = \prod_{m'=0}^{T_p-1} \exp\left[\tau_{\rm EFF}(1)E_{\rm V} - \tau_{\rm EFF}(m)E_T(m-m')\right], \tag{55}$$

where $E_{\rm V}$ is the variational energy. This is necessary in order to avoid floating point errors.

10.9 Growth estimator of the energy

The total weight of a DMC simulation at time $t = m\tau$ is given by:

$$W(t) \equiv \int f(\mathbf{R},t)\,d\mathbf{R} \approx \sum_{\alpha=1}^{N_{\rm config}(m)} w_\alpha(m) \equiv M_{\rm tot}(m). \tag{56}$$

Assuming the DMC simulation to be equilibrated, so the ground state is the only remaining component of the wave function, the time-dependence of the total weight is simply given by [22]:

$$W(t+\tau) = \exp[-(E_0 - E_T)\,\tau]W(t), \tag{57}$$

where E_0 is the (fixed-node) ground-state energy.

Hence the *growth estimator* of the ground state energy for a single iteration is given by:

$$E_0 \approx -\frac{1}{\tau}\log\left(\frac{\exp[-E_T(m+1)\tau]M_{\rm tot}(m+1)}{M_{\rm tot}(m)}\right). \tag{58}$$

By evaluating the expectation value of the argument of the logarithm using the method of UNR [5] and using our estimate of the effective timestep, we obtain a much better estimate of the ground state:

$$\begin{aligned} E_{\rm growth}(m) &= -\frac{1}{\tau_{\rm EFF}(m)}\log\left(\frac{\sum_{m'=1}^{m}\Pi(m',T_p)M_{\rm tot}(m')\left(\frac{\exp[-E_T(m'+1)\tau_{\rm EFF}(m')]M_{\rm tot}(m'+1)}{M_{\rm tot}(m')}\right)}{\sum_{m'=1}^{m}\Pi(m',T_p)M_{\rm tot}(m')}\right) \\ &= -\frac{1}{\tau_{\rm EFF}(m)}\log\left(\frac{\sum_{m'=1}^{m}\Pi(m'+1,T_p+1)M_{\rm tot}(m'+1)}{\sum_{m'=1}^{m}\Pi(m',T_p)M_{\rm tot}(m')}\right). \end{aligned} \tag{59}$$

Equation 59 is used to evaluate the growth estimator of the energy in CASINO if the GROWTH_ESTIMATOR flag is set to .TRUE. in the input file.

Note that when evaluating the growth estimator using Equation 59, one of the constant multiplicative terms in the Π-weights of Equation 55 ($\exp[\tau_{\mathrm{EFF}}(1)E_V]$) does not cancel out. Hence we need to include one such term in the numerator of the argument of the logarithm.

10.10 Automatic block-resetting

Numerous schemes for preventing population control catastrophes due to the occurrence of "persistent electrons" have been investigated. Of these, the one that seems to perform best in practice involves returning to an earlier point in the simulation and changing the random number sequence.

If the TRIP_POPN input variable is set to a non-zero value, then a `config.backup` file will be created. This contains the data in the `config.out` file from the previous block. If the population on a single node exceeds TRIP_POPN, then the data from `config.backup` will be read in, the last block of lines will be erased from the `dmc.hist` and `dmc.hist2` files, and the random number generator will be called a few times so that the configurations go off on new random walks, hopefully avoiding the catastrophe that led to TRIP_POPN being exceeded. (If TRIP_POPN is exceeded in the first or second blocks, then `config.in` will be read in instead of `config.backup`.)

If a block has to be reset more than MAX_REC_ATTEMPTS times then the program will abort with an error.

Great care should be taken when choosing a value for TRIP_POPN. It should be sufficiently large that it cannot interfere with normal population fluctuations: this would lead to population control biasing. (Note that the population often grows rapidly at the start of equilibration: again, it must be ensured that automatic block resetting does not interfere with this natural process.) On the other hand, TRIP_POPN should be sufficiently small that persistence is dealt with quickly and that there is insufficient time for a population of configurations containing a persistent electron to stabilise. Choosing larger block lengths allows the program to return to an earlier point in the simulation, increasing the likelihood that the catastrophe will be avoided.

10.11 Evaluation of orbitals in the determinant part of the wave function

10.11.1 Gaussian basis set

CASINO can handle Gaussian basis sets up to and including angular momentum $l = 4$ (s, p, sp, d, f and g functions). Suppose we have N Gaussians $g_{w=1,\ldots,N}$ located at positions $\mathbf{r}_w$ in the primitive cell. Let $\mathbf{R}$ label the other primitive cells. There are therefore copies of the basic set of Gaussian functions located at positions $\mathbf{r}_w + \mathbf{R}$. We want to evaluate the Bloch orbitals at some point $\mathbf{r}$. The orbital is labelled by band ν and k-point,

$$\phi_{\nu,\mathbf{k}}(\mathbf{r}) = \sum_w C_{\nu,\mathbf{k}} \sum_{\mathbf{R}} g_w(\mathbf{r} - \mathbf{r}_w + \mathbf{R}) \exp[i\mathbf{k}\cdot\mathbf{R}] \; . \tag{60}$$

CASINO implicitly assumes the following about the k-points when in Gaussian mode:

- The k-points form a grid,

$$\mathbf{k}_{lmn} = \frac{l}{q_1}\mathbf{b}_1 + \frac{m}{q_2}\mathbf{b}_2 + \frac{n}{q_3}\mathbf{b}_3 + \mathbf{k}_s \; , \tag{61}$$

  where the q_i are integers, the $\mathbf{b}_i$ are the primitive reciprocal lattice vectors, and $(0 \leq l \leq q_1 - 1)$, $(0 \leq m \leq q_2 - 1)$, $(0 \leq n \leq q_3 - 1)$. In CRYSTAL the offset $\mathbf{k}_s$ is always zero, but CASINO does not assume this.

- CASINO deals only with real orbitals, which can be formed by making linear combinations of the states at $\mathbf{k}$ and $-\mathbf{k}$. The list of k-points in the file gwfn.data must contain only one of each $(\mathbf{k}, -\mathbf{k})$ pair, the presence of the other is assumed. To make a many-body Bloch wave function satisfying the condition that it is multiplied by a phase factor under the replacement $\mathbf{r}_i \rightarrow \mathbf{r}_i + \mathbf{R}_s$, where $\mathbf{R}_s$ is a translation vector of the simulation cell, the offset must satisfy $\mathbf{k}_s = 1/2\mathbf{G}_s$, where $\mathbf{G}_s$ is a reciprocal lattice vector of the simulation cell [12], see Section 10.12.

- Finite systems are treated as if they had a single k-point.

10.11.2 Plane wave basis set

Soon.

10.11.3 Blip function basis set

Soon.

10.11.4 Atomic wave functions on a grid

Soon.

10.11.5 Wannier orbital evaluation

Soon.

10.12 Constructing real orbitals

The Bloch orbitals at an arbitrary point in **k**-space are complex. If the set of wavevectors consists of $\pm\mathbf{k}$ pairs then one can always construct a set of real orbitals spanning the same space as the original complex set. A necessary and sufficient condition for the mesh of Eq. 61 to consist of $\pm\mathbf{k}$ pairs is that $\mathbf{k}_\mathrm{s} = \mathbf{G}_\mathrm{s}/2$, where $\mathbf{G}_\mathrm{s}$ is a reciprocal lattice vector the simulation cell lattice. It is four times more efficient to use real orbitals than complex ones because it takes four multiplications to evaluate the product of two complex numbers but only one multiplication for the product of two real numbers. An orbital satisfying Bloch's theorem can be written as

$$\phi_\mathbf{k}(\mathbf{r}) = u_\mathbf{k}(\mathbf{r})e^{i\mathbf{k}\cdot\mathbf{r}} \ , \tag{62}$$

where $u_\mathbf{k}$ has the periodicity of the primitive lattice. The function $\phi^*_\mathbf{k}$ is a Bloch function with wavevector $-\mathbf{k}$. Therefore we can make two real orbitals from $\phi_\mathbf{k}$ and $\phi^*_\mathbf{k}$ as follows:

$$\begin{aligned} \phi_+(\mathbf{r}) &= \frac{1}{\sqrt{2}}\left[\phi_\mathbf{k}(\mathbf{r}) + \phi^*_\mathbf{k}(\mathbf{r})\right] \ , \\ \phi_-(\mathbf{r}) &= \frac{1}{\sqrt{2}i}\left[\phi_\mathbf{k}(\mathbf{r}) - \phi^*_\mathbf{k}(\mathbf{r})\right] \ . \end{aligned} \tag{63}$$

The orbitals ϕ_+ and ϕ_- are orthogonal if $\phi_\mathbf{k}$ and $\phi^*_\mathbf{k} = \phi_{-\mathbf{k}}$ are orthogonal, which is true unless $\mathbf{k} - (-\mathbf{k}) = \mathbf{G}_\mathrm{p}$, i.e., their wavevectors differ by a reciprocal lattice vector of the primitive lattice. In this case ϕ_+ and $\phi_-\mathbf{k}$ are linearly dependent and we must use only one of them. Therefore the scheme is:

$$\begin{aligned} &\text{Case 1.} \quad \text{If } \mathbf{k} \neq \frac{\mathbf{G}_p}{2} \ \text{ use } \phi_+ \text{ and } \phi_- \ . \\ &\text{Case 2.} \quad \text{If } \mathbf{k} = \frac{\mathbf{G}_p}{2} \ \text{ use } \phi_+ \text{ or } \phi_- \ . \end{aligned} \tag{64}$$

In the second case, if one of ϕ_+ or ϕ_- is zero then obviously one must use the other one.

It may happen that we have in our k-point grid the vectors **k** and $-\mathbf{k}$ which are not related by a reciprocal lattice vector of the primitive lattice. We may wish to occupy only one of the orbitals from these k-points. We can then form a real orbital as $\cos(\theta)\phi_+ + \sin(\theta)\phi_-$, where θ is a phase angle between zero and 2π. If both **k** and $-\mathbf{k}$ orbitals are supplied, CASINO chooses one which in general is sufficient to generate all linearly independent orbitals.

10.13 Cusp corrections for Gaussian orbitals

One of the problems with Gaussian basis sets is that they are unable to describe the cusps in the single-particle orbitals at the nuclei that would be present in the exact HF orbitals, because the Gaussian basis functions have zero gradient at the nuclei on which they are centered. This can lead to considerable difficulties in QMC simulations. In both VMC and DMC methods the energy is calculated as the average over many points in the electron configuration space of the local energy,

$E_L = \Psi^{-1}\hat{H}\Psi$, where $\hat{H}$ is the Hamiltonian and Ψ is the many-electron trial wave function. When an electron approaches a nucleus of charge Z the potential energy contribution to E_L diverges as $-Z/r$, where r is the distance from the nucleus. The kinetic energy operator acting on the cusps in the wave function must therefore supply an equal and opposite divergence in the local kinetic energy, because the local energy is constant everywhere in the configuration space if Ψ is an eigenstate of the Hamiltonian. Unfortunately, when using orbitals expanded in a Gaussian basis set, the kinetic energy is finite at the nucleus and therefore E_L diverges. In practice one finds that the local energy has wild oscillations close to the nucleus, which give rise to a large variance in the energy. This is undesirable in VMC, but within DMC it can lead to severe bias and even to catastrophic numerical instabilities.

It might seem that using local basis functions which can be made individually to obey the cusp conditions at the nucleus on which they are centered, such as a Slater-type basis of cusped exponential functions, would solve the problem. This is only partially true however. One issue is the fact that the codes available for performing calculations with Slater-type functions are more limited and less widely used than those which employ Gaussian basis sets. The more important point is that the use of a Slater basis in which each basis function obeys the cusp conditions at the nucleus on which it is centered does *not* automatically lead to a full solution of the electron–nucleus cusp problem. This is due to the contributions from the tails of basis functions centered on other nuclei, which in general prevent molecular orbitals expanded in that basis from obeying the electron–nucleus cusp condition. Manten and Lüchow [16] have developed a scheme for applying cusp corrections in QMC calculations but, as it similarly relies on correcting individual atom-centered basis functions, it is clearly not a full solution.

An alternative solution to the cusp problem might be to enforce the electron–nucleus cusp condition using the Jastrow factor. This is feasible and we have implemented it, but we found this to be unsatisfactory because a very large number of variable parameters are required in order to obtain a good trial wave function.

The solution we have finally adopted in CASINO involves the direct modification of the molecular orbitals so that each of them obeys the cusp condition at each nucleus. This ensures that the local energy remains finite whenever an electron is in the vicinity of a nucleus, although it generally has a discontinuity at the nucleus.

10.13.1 Electron–nucleus cusp corrections

The Kato cusp condition [42] applied to an electron at $\mathbf{r}_i$ and a nucleus of charge Z at the origin is

$$\left(\frac{\partial\langle\Psi\rangle}{\partial r_i}\right)_{r_i=0} = -Z\langle\Psi\rangle_{r_i=0} \,, \tag{65}$$

where $\langle\Psi\rangle$ is the spherical average of the many-body wave function about $\mathbf{r}_i = 0$. For a determinant of orbitals to obey the Kato cusp condition at the nuclei it is sufficient for every orbital to obey Eq. (65) at every nucleus. We need only correct the orbitals which are non-zero at a particular nucleus because the others already obey Eq. (65). This is sufficient to guarantee that the local energy is finite at the nucleus provided at least one orbital is non-zero there. In the unlikely case that all of the orbitals are zero at the nucleus then the probability of an electron being at the nucleus is zero and it is not important whether Ψ obeys the cusp condition.

An orbital, ψ, expanded in a Gaussian basis set can be written as

$$\psi = \phi + \eta \,, \tag{66}$$

where ϕ is the part of the orbital arising from the s-type Gaussian functions centered on the nucleus in question (which, for convenience is at $\mathbf{r} = 0$), and η is the rest of the orbital. The spherical average of ψ about $\mathbf{r} = 0$ is given by

$$\langle\psi\rangle = \phi + \langle\eta\rangle \,. \tag{67}$$

In our scheme we seek a corrected orbital, $\tilde{\psi}$, which differs from ψ only in the part arising from the s-type Gaussian functions centered on the nucleus, i.e.,

$$\tilde{\psi} = \tilde{\phi} + \eta \,. \tag{68}$$

The correction, $\tilde{\psi} - \psi$, is therefore spherically symmetric about the nucleus. We now demand that $\tilde{\psi}$ obeys the cusp condition at $\mathbf{r} = 0$,

$$\left(\frac{d\langle \tilde{\psi} \rangle}{dr} \right)_0 = -Z \langle \tilde{\psi} \rangle_0 \ . \tag{69}$$

Note that $\langle \eta \rangle$ is cusp-less because it arises from the Gaussian basis functions centered on the origin with non-zero angular momentum, whose spherical averages are zero, and the tails of the Gaussian basis functions centered on other sites, which must be cusp-less at the nucleus in question. We therefore obtain

$$\left(\frac{d\tilde{\phi}}{dr} \right)_0 = -Z \left(\tilde{\phi}(0) + \eta(0) \right) \ . \tag{70}$$

We use Eq. (70) as the basis of our scheme for constructing cusp-corrected orbitals. We have found that it is important to include $\eta(0)$ when correcting molecular orbitals expanded in Gaussian basis functions. We suspect this will also be so for Slater-type functions, and our cusp correction scheme could also be applied in that case to impose the cusp conditions exactly.

10.13.2 Cusp correction algorithm

One could conceive of correcting the orbitals either by adding a function to the Gaussian orbital inside some reasonably small radius, multiplying by a function (e.g., using the Jastrow factor as mentioned earlier, or by replacing the orbital near the nucleus by a function which obeys the cusp condition. However, as the local energy obtained from Gaussian orbitals shows wild oscillations close to the nucleus, the best option seems to be the latter one: replacement of the orbital inside some small radius by a well-behaved form.

We apply a cusp correction to each orbital at each nucleus at which it is non-zero. Inside some cusp correction radius r_c we replace ϕ, the part of the orbital arising from s-type Gaussian functions centered on the nucleus in question, by

$$\tilde{\phi} = C + \mathrm{sgn}[\tilde{\phi}(0)] \exp[p(r)] = C + R(r). \tag{71}$$

In this expression $\mathrm{sgn}[\tilde{\phi}(0)]$ is ± 1, reflecting the sign of $\tilde{\phi}$ at the nucleus, and C is a shift chosen so that $\tilde{\phi} - C$ is of one sign within r_c. This shift is necessary since the uncorrected s-part of the orbital ϕ may have a node where it changes sign inside the cusp correction radius, and we wish to replace ϕ by an exponential function, which is necessarily of one sign everywhere. The polynomial p is given by

$$p = \alpha_0 + \alpha_1 r + \alpha_2 r^2 + \alpha_3 r^3 + \alpha_4 r^4 \ , \tag{72}$$

and we determine α_0, α_1, α_2, α_3, and α_4 by imposing five constraints on $\tilde{\phi}$. We demand that the value and the first and second derivatives of $\tilde{\phi}$ match those of the s-part of the Gaussian orbital at $r = r_c$. We also require that the cusp condition is satisfied at $r = 0$. We use the final degree of freedom to optimize the behavior of the local energy in a manner to be described below. However, if we impose such a constraint directly the equations satisfied by the α_i cannot be solved analytically. This is inconvenient and we found that a superior algorithm was obtained by imposing a fifth constraint which allows the equations to be solved analytically, and then searching over the value of the fifth constraint for a "good solution". To this end we chose to constrain the value of $\tilde{\phi}(0)$. With these constraints we have:

1.
$$\ln |\tilde{\phi}(r_c) - C| = p(r_c) = X_1; \tag{73}$$

2.
$$\left. \frac{1}{R(r_c)} \frac{d\tilde{\phi}}{dr} \right|_{r_c} = p'(r_c) = X_2; \tag{74}$$

3.
$$\left. \frac{1}{R(r_c)} \frac{d^2\tilde{\phi}}{dr^2} \right|_{r_c} = p''(r_c) + p'^2(r_c) = X_3; \tag{75}$$

4.

$$\frac{1}{R(0)}\frac{d\tilde{\phi}}{dr}\bigg|_0 = p'(0) = -Z\left(\frac{C+R(0)+\eta(0)}{R(0)}\right) = X_4; \tag{76}$$

5.

$$\ln|\tilde{\phi}(0) - C| = p(0) = X_5. \tag{77}$$

Although the constraint equations are non-linear, they can be solved analytically, giving

$$\begin{aligned}
\alpha_0 &= X_5 \\
\alpha_1 &= X_4 \\
\alpha_2 &= 6\frac{X_1}{r_c^2} - 3\frac{X_2}{r_c} + \frac{X_3}{2} - 3\frac{X_4}{r_c} - 6\frac{X_5}{r_c^2} - \frac{X_2^2}{2} \\
\alpha_3 &= -8\frac{X_1}{r_c^3} + 5\frac{X_2}{r_c^2} - \frac{X_3}{r_c} + 3\frac{X_4}{r_c^2} + 8\frac{X_5}{r_c^3} + \frac{X_2^2}{r_c} \\
\alpha_4 &= 3\frac{X_1}{r_c^4} - 2\frac{X_2}{r_c^3} + \frac{X_3}{2r_c^2} - \frac{X_4}{r_c^3} - 3\frac{X_5}{r_c^4} - \frac{X_2^2}{2r_c^2}.
\end{aligned} \tag{78}$$

Our procedure is to solve Eq. (78) using an initial value of $\tilde{\phi}(0) = \phi(0)$. We then vary $\tilde{\phi}(0)$ so that the "effective one-electron local energy",

$$\begin{aligned}
E_L^s(r) &= \tilde{\phi}^{-1}\left[-\frac{1}{2}\nabla^2 - \frac{Z_{\text{eff}}}{r}\right]\tilde{\phi} \\
&= -\frac{1}{2}\frac{R(r)}{C+R(r)}\left[\frac{2p'(r)}{r} + p''(r) + p'^2(r)\right] - \frac{Z_{\text{eff}}}{r},
\end{aligned} \tag{79}$$

is well-behaved. Here the effective nuclear charge Z_{eff} is given by

$$Z_{\text{eff}} = Z\left(1 + \frac{\eta(0)}{C+R(0)}\right), \tag{80}$$

which ensures that $E_L^s(0)$ is finite when the cusp condition of Eq. (76) is satisfied.

We studied the effective one-electron local energies obtained using Eq. (79) with $Z_{\text{eff}} = Z$ for the $1s$ and $2s$ all-electron Hartree-Fock orbitals of neutral atoms calculated by numerical integration on fine radial grids for atoms up to $Z = 82$. We noticed that the quantity $E_L^s(r)/Z^2$ is only weakly dependent on Z in the range $r < 1.5/Z$. We therefore chose an "ideal" effective one-electron local energy curve given by

$$\begin{aligned}
\frac{E_L^{\text{ideal}}(r)}{Z^2} &= \beta_0 + \beta_1 r^2 + \beta_2 r^3 + \beta_3 r^4 \\
&\quad + \beta_4 r^5 + \beta_5 r^6 + \beta_6 6r^7 + \beta_7 r^8.
\end{aligned} \tag{81}$$

The values chosen for the coefficients were $\beta_1 = 3.25819$, $\beta_2 = -15.0126$, $\beta_3 = 33.7308$, $\beta_4 = -42.8705$, $\beta_5 = 31.2276$, $\beta_6 = -12.1316$, $\beta_7 = 1.94692$, obtained by fitting to the data for the $1s$ orbital of the carbon atom. The value of β_0 depends on the particular atom and its environment. The ideal effective one-electron local energy for a particular orbital is chosen to have the functional form of $E_L^{\text{ideal}}(r)$, but with the constant value β_0 chosen so that the effective one-electron local energy is continuous at r_c. Hydrogen is treated as a special case as the $1s$ orbital of the isolated atom is only half-filled, and we use $E_L^{\text{ideal}}(r) = \beta_0 - 0.5$.

We wish to choose $\tilde{\phi}(0)$ so that $E_L^s(r)$ is as close as possible to $E_L^{\text{ideal}}(r)$ for $0 < r < r_c$, i.e., the effective one-electron local energy is required to follow the "ideal" curve as closely as possible. In our current implementation we find the best $\tilde{\phi}(0)$ by minimizing the maximum square deviation from the ideal energy, $[E_L^s(r) - E_L^{\text{ideal}}(r)]^2$, within this range. Beginning with $\tilde{\phi}(0) = \phi(0)$, we first bracket the minimum then refine $\tilde{\phi}(0)$ using a simple golden section search. In principle we are more interested in $E_L^s(r)$ being close to $E_L^{\text{ideal}}(r)$ near r_c than near zero because the probability of an electron being near r_c is normally much greater than it being near the nucleus. One might therefore consider using a

weighting factor and minimizing, e.g., $[r(E_L^s(r) - E_L^{\rm ideal}(r))]^2$. In practical calculations this was found not to improve the result in general and weighting factors were not used in our final implementation.

It is clearly important to find an automatic procedure for choosing appropriate values of the cusp correction radii. Although the final quality of the wave function in QMC calculations is expected to have only a relatively weak dependence on its precise value, the optimal cusp correction radius r_c for each orbital and nucleus should depend on the quality of the basis set and on the shape of the orbital in question. In particular one would expect the cusp correction radii to become smaller as the quality of the basis set is improved. Although clearly many other schemes are possible, we choose the r_c in our implementation as follows. The maximum possible cusp correction radius is taken to be $r_{c,max} = 1/Z$. The actual value of r_c is then determined by a universal parameter c_c ('CUSP_CONTROL' in the CASINO input file) for which a default value of 50 was found to be reasonable. The cusp correction radius r_c for each orbital and nucleus is set equal to the largest radius less than $r_{c,max}$ at which the deviation of the effective one-electron local energy calculated with ϕ from the ideal curve has a magnitude greater than Z^2/c_c. Appropriate polynomial coefficients α_i and the resulting maximum deviation of the effective one-electron local energy from the ideal curve are then calculated for this r_c. As a final refinement one might then allow the code to vary r_c over a relatively small range centered on the initial value, recomputing the optimal polynomial cusp correction at each radius, in order to optimize further the behavior of the effective one-electron local energy. This is done by default in the implementation.

When a Gaussian orbital can be readily identified as, for example, a $1s$ orbital, it generally does not have a node within $r_{c,max}$. In many cases, however, some of the molecular orbitals have small s-components which may have nodes close to the nucleus. The possible presence of nodes inside the cusp correction radius complicates the procedure because the effective one-electron local energy diverges there. One could simply force the cusp correction radius to be less than the radius of the node closest to the nucleus, but in practice nodes can be very close to the nucleus and such a constraint severely restricts the flexibility of the algorithm. In practice we define small regions around each node where the effective one-electron local energies are not taken into account during the minimization, and from which the cusp correction radius is excluded.

Gaussian cusp correction is activated through the input keyword CUSP_CORRECTION - this has an effect only if the system contains at least one all-electron atom. In periodic systems it has proved difficult to implement this scheme efficiently, and while it works perfectly well the performance of the code is significantly affected. Check the timings. More information about the cusp correction of each orbital at each nucleus can be produced with the keyword CUSP_INFO which can be useful in fine tuning. Clearly this can produce a lot of output so beware.

10.14 The Jastrow factor

The Slater-Jastrow wave function is

$$\Psi(\mathbf{R}) = \exp[J] \sum_n c_n D_n^{\uparrow} D_n^{\downarrow} \, , \tag{82}$$

where the D_n are determinants of up and down spin orbitals and J is the Jastrow factor.

10.14.1 General form of CASINO's new Jastrow factor

CASINO's new Jastrow factor is the sum of homogeneous, isotropic electron–electron terms u, isotropic electron–nucleus terms χ centred on the nuclei, isotropic electron–electron–nucleus terms f, also centred on the nuclei and, in periodic systems, plane-wave expansions of electron–electron separation and electron position, p and q. The form is

$$\begin{aligned} J(\{\mathbf{r}_i\},\{\mathbf{r}_I\}) \;=\; & \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} u(r_{ij}) + \sum_{I=1}^{N_{\rm ions}}\sum_{i=1}^{N}\chi_I(r_{iI}) + \sum_{I=1}^{N_{\rm ions}}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_I(r_{iI}, r_{jI}, r_{ij}) \\ & + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} p(\mathbf{r}_{ij}) + \sum_{i=1}^{N} q(\mathbf{r}_i), \end{aligned} \tag{83}$$

where N is the number of electrons, $N_{\rm ions}$ is the number of ions, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $\mathbf{r}_{iI} = \mathbf{r}_i - \mathbf{r}_I$, $\mathbf{r}_i$ is the position of electron i and $\mathbf{r}_I$ is the position of nucleus I. In periodic systems the electron–electron

and electron–ion separations, $\mathbf{r}_{ij}$ and $\mathbf{r}_{iI}$, are evaluated under the minimum-image convention. Note that u, χ, f, p and q may also depend on the spins of electrons i and j.

The plane-wave term, p, will describe similar sorts of correlation to the u term. In periodic systems the u term must be cut off at a distance less than or equal to the Wigner–Seitz radius of the simulation cell and therefore the u function includes electron pairs over less than three quarters of the simulation cell. The p term adds variational freedom in the "corners" of the simulation cell, which could be important in small cells. The p term can also describe anisotropic correlations, such as might be encountered in a layered compound. It is expected that the u term will be considerably more important than the p term, which cannot describe the electron–electron cusps and is therefore best limited to describing longer-ranged correlations. The q term will describe similar electron–nucleus correlations to the χ_I terms.

10.14.2 The u, χ and f terms in the new Jastrow factor

The u term consists of a complete power expansion in r_{ij} up to order $r_{ij}^{C+N_u}$ which satisfies the Kato cusp conditions at $r_{ij} = 0$, goes to zero at the cutoff length, $r_{ij} = L_u$, and has $C - 1$ continuous derivatives at L_u:

$$u(r_{ij}) = (r_{ij} - L_u)^C \times \Theta(L_u - r_{ij}) \times \left(\alpha_0 + \left[\frac{\Gamma_{ij}}{(-L_u)^C} + \frac{\alpha_0 C}{L_u} \right] r_{ij} + \sum_{l=2}^{N_u} \alpha_l r_{ij}^l \right), \tag{84}$$

where Θ is the Heaviside function and $\Gamma_{ij} = 1/2$ if electrons i and j have opposite spins and $\Gamma_{ij} = 1/4$ if they have the same spin. In this expression C determines the behaviour at the cutoff length. If $C = 2$, the gradient of u is continuous but the second derivative and hence the local energy is discontinuous, and if $C = 3$ then both the gradient of u and the local energy are continuous.

The form of χ is

$$\chi_I(r_{iI}) = (r_{iI} - L_{\chi I})^C \times \Theta(L_{\chi I} - r_{iI}) \times \left(\beta_{0I} + \left[\frac{-Z_I}{(-L_{\chi I})^C} + \frac{\beta_{0I} C}{L_{\chi I}} \right] r_{iI} + \sum_{m=2}^{N_\chi} \beta_{mI} r_{iI}^m \right). \tag{85}$$

It may be assumed that $\beta_{mI} = \beta_{mJ}$ where I and J are equivalent ions. The term involving the ionic charge Z_I enforces the electron–nucleus cusp condition.

The expression for f is the most general expansion of a function of r_{ij}, r_{iI} and r_{jI} that does not interfere with the Kato cusp conditions and goes smoothly to zero when either r_{iI} or r_{jI} reach cutoff lengths:

$$f_I(r_{iI}, r_{jI}, r_{ij}) = (r_{iI} - L_{fI})^C (r_{jI} - L_{fI})^C \Theta(L_{fI} - r_{iI}) \Theta(L_{fI} - r_{jI}) \sum_{l=0}^{N_{fI}^{\mathrm{eN}}} \sum_{m=0}^{N_{fI}^{\mathrm{eN}}} \sum_{n=0}^{N_{fI}^{\mathrm{ee}}} \gamma_{lmnI} r_{iI}^l r_{jI}^m r_{ij}^n. \tag{86}$$

Various restrictions are placed on γ_{lmnI}. To ensure the Jastrow factor is symmetric under electron exchanges it is demanded that $\gamma_{lmnI} = \gamma_{mlnI}\ \forall\, I, m, l, n$. If ions I and J are equivalent then it is demanded that $\gamma_{lmnI} = \gamma_{lmnJ}$. The condition that the f term has no electron–electron cusps is

$$\left(\frac{\partial f}{\partial r_{ij}} \right)_{\substack{r_{ij}=0 \\ r_{iI}=r_{jI}}} = 0, \tag{87}$$

which implies that

$$\sum_{l=0}^{N_{fI}^{\mathrm{eN}}} \sum_{m=0}^{N_{fI}^{\mathrm{eN}}} \gamma_{lm1I} r_{iI}^{l+m} (r_{iI} - L_{fI})^{2C} = 0\ , \tag{88}$$

for all r_{iI}. Hence, $\forall\, k \in \{0, \ldots, 2N_{fI}^{\mathrm{eN}}\}$,

$$\sum_{l,m\ :\ l+m=k} \gamma_{lm1I} = 0. \tag{89}$$

The condition that the f term has no electron–nucleus cusps is

$$\left(\frac{\partial f}{\partial r_{iI}}\right)_{\substack{r_{iI}=0 \\ r_{ij}=r_{jI}}} = 0, \tag{90}$$

which gives

$$\sum_{m=0}^{N_{fI}^{\mathrm{eN}}} \sum_{n=0}^{N_{fI}^{\mathrm{ee}}} (C\gamma_{0mnI} - L_{fI}\gamma_{1mnI})(-L_{fI})^{C-1} r_{jI}^{m+n} (r_{jI} - L_{fI})^{C} = 0, \tag{91}$$

for all r_{jI}. It is therefore required that, $\forall\, k' \in \{0, \ldots, N_{fI}^{\mathrm{eN}} + N_{fI}^{\mathrm{ee}}\}$,

$$\sum_{m,n\,:\,m+n=k'} (C\gamma_{0mnI} - L_{fI}\gamma_{1mnI}) = 0. \tag{92}$$

10.14.3 The p and q terms in the new Jastrow factor

The p term takes the cuspless form

$$p(\mathbf{r}_{ij}) = \sum_A a_A \sum_{\mathbf{G}_A^+} \cos(\mathbf{G}_A \cdot \mathbf{r}_{ij}) \ , \tag{93}$$

where the $\{\mathbf{G}_A\}$ are the reciprocal lattice vectors of the simulation cell belonging to the Ath star of vectors that are equivalent under the full symmetry group of the Bravais lattice, and "+" means that, if $\mathbf{G}_A$ is included in the sum, $-\mathbf{G}_A$ is excluded.

For systems with inversion symmetry the q term takes the cuspless form

$$q(\mathbf{r}_i) = \sum_B b_B \sum_{\mathbf{G}_B^+} \cos(\mathbf{G}_B \cdot \mathbf{r}_i), \tag{94}$$

where the $\{\mathbf{G}_B\}$ are the reciprocal lattice vectors of the primitive unit cell belonging to the Bth star of vectors that are equivalent under the space-group symmetry of the crystal, and the "+" means that, if $\mathbf{G}_B$ is included in the sum, $-\mathbf{G}_B$ is excluded.

10.14.4 Old Jastrow factor

The old redundant Jastrow factor `jasfun.data` used by CASINO before Easter 2004 is described in this section (as it is still supported) can be written as follows (We denote the distance of electron i from ion I by r_{iI} and the distance between electrons i and j by r_{ij}).

$$\begin{aligned} J = \ & -\sum_{i>j,j} [u_0(r_{ij}) + S_1(r_{ij})] - \sum_i \sum_I S_2(r_{iI}) \\ & - \sum_{i>j,j} \sum_I [S_3(r_{iI}, r_{jI}) + S_4(r_{iI}, r_{ij}) + S_4(r_{jI}, r_{ij}) + S_5(r_{iI}, r_{jI}, r_{ij})] \ . \end{aligned} \tag{95}$$

The one-electron term S_2 is allowed to depend on the electron spin, while the other terms which all involve two electrons are allowed to depend on the relative spins. At the moment the terms for two up-spin electrons are the same as for two down-spin electrons. In three dimensions, the function u_0 is taken to be

$$u_0(r_{ij}) = \frac{A}{r_{ij}} \left[1 - \exp\left(-\frac{r_{ij}}{F}\right)\right] \exp\left(-\frac{r_{ij}^2}{L_0^2}\right) \ , \tag{96}$$

where F is chosen so that the cusp conditions are obeyed, i.e., $F_{\uparrow\uparrow} = \sqrt{2A}$ and $F_{\uparrow\downarrow} = \sqrt{A}$. L_0 is chosen so that u_0 is effectively zero at the shortest distance to the surface of the Wigner-Seitz cell, L_{WS}. We normally take $L_0 \approx 0.3 L_{\mathrm{WS}}$, and A is either set to the inverse of the plasma energy or obtained by an optimization procedure. In two-dimensional electron and electron-hole gases, u_0 is set to

$$u_0(r_{ij}) = \frac{A}{\sqrt{r_{ij}}} \left[1 - \exp\left(-\frac{r_{ij}}{2F} - \sqrt{\frac{r_{ij}}{F}}\right)\right] \exp\left(-\frac{r_{ij}^2}{L_0^2}\right) \ , \tag{97}$$

where cusp conditions make $F_{\uparrow\uparrow} = A^{2/3}$ and $F_{\uparrow\downarrow} = (A/3)^{2/3}$.

The functions S_1 to S_5 are

$$S_1 = (r_{ij} - L')^2 r_{ij}^2 \sum_{l=0}^{l1} \alpha_l T_l(\bar{r}'_{ij}) + B'(r_{ij} - L')^2 \left(\frac{L'}{2} + r_{ij} \right) \tag{98}$$

$$S_2 = (r_i - L)^2 r_i^2 \sum_{l=0}^{l2} \beta_l T_l(\bar{r}_i) + B(r_i - L)^2 \left(\frac{L}{2} + r_i \right) \tag{99}$$

$$S_3 = (r_i - L)^2 (r_j - L)^2 r_i^2 r_j^2 \sum_{l=0}^{lm3} \sum_{m=0}^{lm3} \gamma_{lm} T_l(\bar{r}_i) T_m(\bar{r}_j) \tag{100}$$

$$S_4 = (r_i - L)^2 (r_{ij} - 2L)^2 r_i^2 r_{ij}^2 \sum_{l=0}^{l4} \sum_{m=0}^{m4} \epsilon_{lm} T_l(\bar{r}_i) T_m(\bar{r}_{ij}) \tag{101}$$

$$S_5 = (r_i - L)^2 (r_j - L)^2 r_i^2 r_j^2 r_{ij}^2 \sum_{l=0}^{lm5} \sum_{m=0}^{lm5} \sum_{n=0}^{nm5} \omega_{lmn} T_l(\bar{r}_i) T_m(\bar{r}_j) T_n(\bar{r}_{ij}) \ , \tag{102}$$

where T_l denotes the lth Chebyshev polynomial and the expansion range is rescaled to the orthogonality interval of the Chebyshev polynomials, [-1,1], so that

$$\bar{r}'_{ij} = \frac{2r_{ij} - L'}{L'} \tag{103}$$

$$\bar{r}_i = \frac{2r_i - L}{L} \tag{104}$$

$$\bar{r}_{ij} = \frac{r_{ij} - L}{L} \ . \tag{105}$$

We assume that terms are zero according to:

$$S_1 = 0 \ \text{ when } \ r_{ij} > L' \tag{106}$$

$$S_2 = 0 \ \text{ when } \ r_i > L \tag{107}$$

$$S_3 = 0 \ \text{ when } \ r_i > L \ \text{ or } \ r_j > L \tag{108}$$

$$S_4 = 0 \ \text{ when } \ r_i > L \ \text{ or } \ r_{ij} > 2L \tag{109}$$

$$S_5 = 0 \ \text{ when } \ r_i > L \ \text{ or } \ r_j > L \ . \tag{110}$$

To maintain the even symmetry of the Jastrow factor under interchange of the positions of two electrons, γ_{lm} and ω_{lmn} are forced to be symmetric under interchange of l and m.

The parameters in S_1 to S_5 are obtained from variance minimization. The computational cost of including terms other than u_0, S_1 and S_2 can be large. S_2 is chosen to be cusp-less as we assume that the ionic potentials are finite at the ion centre. Because u_0 is chosen to obey the cusp electron-electron conditions, S_1, S_3, S_4 and S_5 are chosen to be cusp-less. L is the range of the short distance correlation functions, which is normally chosen to be of order a bond length and L' is the range of the long distance correlation function, which is normally chosen to be the Wigner-Seitz radius for a periodic system or approximately the size of the system for finite systems. Functions which depend on the distances to ions can be chosen to be different for different "sets" of ions. The most important terms are u_0 and S_1 which give a homogeneous "u" function, and S_2, which amounts to a "χ" function.

10.15 Wave function updating

Consider a Slater-Jastrow wave function

$$\Psi(\mathbf{R}) = e^{J(\mathbf{R})} D^{\uparrow}(\mathbf{r}_1, \ldots, \mathbf{r}_{N_\uparrow}) D^{\downarrow}(\mathbf{r}_{N_\uparrow+1}, \ldots, \mathbf{r}_N) \ , \tag{111}$$

where J is the Jastrow factor and $D^{\uparrow}$ and $D^{\downarrow}$ are Slater determinants for the spin-up and spin-down electrons respectively. We will need to calculate the ratio of the new wave function to the old when,

for example, the ith spin-up electron is moved from $\mathbf{r}_i^{\rm old}$ to $\mathbf{r}_i^{\rm new}$. The wave function ratio can be written as

$$\frac{\Psi(\mathbf{R}^{\rm new})}{\Psi(\mathbf{R}^{\rm old})} = q^{\uparrow}\frac{e^J(\mathbf{R}^{\rm new})}{e^J(\mathbf{R}^{\rm old})} \,, \tag{112}$$

where

$$q^{\uparrow} = \frac{D^{\uparrow}(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_i^{\rm new},\ldots,\mathbf{r}_N)}{D^{\uparrow}(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_i^{\rm old},\ldots,\mathbf{r}_N)} \,. \tag{113}$$

The contribution from the Jastrow factor is easily evaluated. A direct calculation of the determinants in $q^{\uparrow}$ at every move by LU decomposition is time-consuming, and instead we use an updating method. We define the Slater matrix $\mathcal{D}^{\uparrow}$ via

$$\mathcal{D}^{\uparrow}_{jk} = \psi_j(\mathbf{r}_k) \,, \tag{114}$$

where ψ_j is the jth one-electron orbital of the spin-up Slater determinant and $\mathbf{r}_k$ is the position of the kth spin-up electron. The transpose of the inverse of $\mathcal{D}^{\uparrow}$, which we call $\overline{\mathcal{D}}^{\uparrow}$, may be expressed in terms of the cofactors and determinant of $\mathcal{D}^{\uparrow}$,

$$\overline{\mathcal{D}}^{\uparrow}_{jk} = \frac{\mathrm{cof}(\mathcal{D}^{\uparrow}_{jk})}{\det(\mathcal{D}^{\uparrow})} \,. \tag{115}$$

The move of electron i changes only the ith column of $\mathcal{D}^{\uparrow}$ and so does not affect any of the cofactors associated with this column. The new Slater determinant may be expanded in terms of these cofactors and the result divided by the old Slater determinant to obtain

$$q^{\uparrow} = \frac{\det(\mathcal{D}^{\uparrow,\rm new})}{\det(\mathcal{D}^{\uparrow,\rm old})} = \sum_j \psi_j(\mathbf{r}_i^{\rm new})\overline{\mathcal{D}}^{\uparrow,\rm old}_{ji} \,. \tag{116}$$

If the $\overline{\mathcal{D}}^{\uparrow}$ matrix is known, one can compute $q^{\uparrow}$ in a time proportional to N.

Evaluating $\overline{\mathcal{D}}^{\uparrow}$ using LU decomposition takes of order N^3 operations; but once the initial $\overline{\mathcal{D}}^{\uparrow}$ matrix has been calculated it can be updated at a cost proportional to N^2 using the formulae

$$\overline{\mathcal{D}}^{\uparrow,\rm new}_{ji} = \frac{1}{q^{\uparrow}}\overline{\mathcal{D}}^{\uparrow,\rm old}_{ji} \,, \tag{117}$$

in the case where $k = i$, and

$$\overline{\mathcal{D}}^{\uparrow,\rm new}_{jk} = \overline{\mathcal{D}}^{\uparrow,\rm old}_{jk} - \frac{1}{q^{\uparrow}}\overline{\mathcal{D}}^{\uparrow,\rm old}_{ji}\sum_m \psi_m(\mathbf{r}_i^{\rm new})\overline{\mathcal{D}}^{\uparrow,\rm old}_{mk} \,, \tag{118}$$

when $k \neq i$.

10.16 Evaluating the local energy

The local energy is given by

$$E_L(\mathbf{R}) = \sum_{i=1}^{N} -\frac{1}{2}\Psi^{-1}(\mathbf{R})\nabla_i^2\Psi(\mathbf{R}) + \sum_{i=1}^{N} V(\mathbf{R}) + \sum_{i=1}^{N}\Psi^{-1}(\mathbf{R})\hat{V}^{\rm ps}_{{\rm nl},i}\Psi(\mathbf{R}) + V_{\rm CPP}(\mathbf{R}) + \sum_{i>j}^{N} v_{\rm e-e}(\mathbf{R}) \,, \tag{119}$$

where the terms are the kinetic energy, the local part of the external potential energy, the non-local part of the potential energy, the core polarization potential energy (if present), and the electron-electron interaction energy. The evaluation of the kinetic energy is discussed in Section 10.17. The evaluation of the non-local energy is discussed in Section 10.18, while the core polarization potential energy is discussed in Section 10.19. The local part of the external potential energy is divided into a short range part around each ion, which is evaluated directly, and a long range Coulomb part which is evaluated using the Ewald potential in periodic systems, see Section 10.20, or simply as a sum of $1/r$ potentials in finite systems. The electron-electron interaction energy is evaluated either using the Ewald interaction or the MPC interaction, see Section 10.20.4 in periodic systems, or simply as a sum of $1/r$ potentials in finite systems.

10.17 Evaluating the kinetic energy

The kinetic part of the local energy, K, can be expressed as a sum of contributions from each electron,

$$K = \sum_{i=1}^{N} K_i = \sum_{i=1}^{N} -\frac{1}{2}\Psi(\mathbf{R})^{-1}\nabla_i^2\Psi(\mathbf{R}) \; . \tag{120}$$

Because of the exponential form of the Jastrow factor, it is convenient to re-express K_i in terms of the logarithm of Ψ. We define

$$T_i = -\frac{1}{4}\nabla_i^2 \left(\ln|\Psi|\right) = -\frac{1}{4}\frac{\nabla_i^2\Psi}{\Psi} + \frac{1}{4}\left(\frac{\nabla_i\Psi}{\Psi}\right)^2 \; , \tag{121}$$

and the drift vector $\mathbf{F}_i$,

$$\mathbf{F}_i = \frac{1}{\sqrt{2}}\nabla_i \left(\ln|\Psi|\right) = \frac{1}{\sqrt{2}}\frac{\nabla_i\Psi}{\Psi} \; . \tag{122}$$

Therefore

$$K_i = 2T_i - |\mathbf{F}_i|^2 \; . \tag{123}$$

In VMC an integration by parts shows that

$$\langle K\rangle = \langle|\mathbf{F}|^2\rangle = \langle T\rangle \; , \tag{124}$$

where the angle brackets denote averages over the variational distribution, $|\Psi(\mathbf{R})|^2$. Eq. (124) provides a useful consistency check for VMC calculations but note that it does not hold exactly within DMC, except in the limit of perfect importance sampling. In VMC the kinetic energy may be evaluated using any of the three estimators in Eq. (124). CASINO automatically uses $\langle K\rangle$ for the evaluation of the total energy, because this normally leads to the lowest variance. However, the lowest variance of the kinetic energy itself is often obtained from $\langle T\rangle$. In DMC the three estimators of Eq. (124) are not exactly equivalent and $\langle K\rangle$ should always be used.

For the Slater-Jastrow wave function of Eq. (111) we have

$$\nabla_i \left(\ln|\Psi|\right) = \frac{\nabla_i D^{\sigma_i}}{D^{\sigma_i}} + \nabla_i J \; , \tag{125}$$

$$\nabla_i^2 \left(\ln|\Psi|\right) = \frac{\nabla_i^2 D^{\sigma_i}}{D^{\sigma_i}} - \left(\frac{\nabla_i D^{\sigma_i}}{D^{\sigma_i}}\right)^2 + \nabla_i^2 J \; . \tag{126}$$

The terms involving Slater determinants may be evaluated by expanding D^{σ_i} in terms of the cofactors of the ith column of the Slater matrix $\mathcal{D}^{\sigma_i}$. If electron i has spin up, for example, the required expansion is

$$D^{\uparrow} = \det\left(\mathcal{D}^{\uparrow}\right) = \sum_j \psi_j(\mathbf{r}_i)\,\mathrm{cof}(\mathcal{D}^{\uparrow}_{ji}) \; . \tag{127}$$

Since all the cofactors appearing in this equation are independent of $\mathbf{r}_i$, we obtain

$$\frac{\nabla_i D^{\uparrow}}{D^{\uparrow}} = \sum_j \left(\nabla_i\psi_j(\mathbf{r}_i)\right)\overline{\mathcal{D}}^{\uparrow}_{ji} \; , \tag{128}$$

$$\frac{\nabla_i^2 D^{\uparrow}}{D^{\uparrow}} = \sum_j \left(\nabla_i^2\psi_j(\mathbf{r}_i)\right)\overline{\mathcal{D}}^{\uparrow}_{ji} \; . \tag{129}$$

When moving electron i from $\mathbf{r}_i^{\mathrm{old}}$ to $\mathbf{r}_i^{\mathrm{new}}$, it is useful to be able to evaluate the kinetic energy at the new position before updating the $\overline{\mathcal{D}}$ matrix. Since the cofactors in Eq. (127) are independent of $\mathbf{r}_i$, Eqs. (128) and (129) become

$$\frac{\nabla_i D^{\uparrow,\mathrm{new}}}{D^{\uparrow,\mathrm{new}}} = \frac{1}{q^{\uparrow}}\sum_j \left(\nabla_i\psi_j(\mathbf{r}_i^{\mathrm{new}})\right)\overline{\mathcal{D}}^{\uparrow,\mathrm{old}}_{ji} \; , \tag{130}$$

$$\frac{\nabla_i^2 D^{\uparrow,\mathrm{new}}}{D^{\uparrow,\mathrm{new}}} = \frac{1}{q^{\uparrow}}\sum_j \left(\nabla_i^2\psi_j(\mathbf{r}_i^{\mathrm{new}})\right)\overline{\mathcal{D}}^{\uparrow,\mathrm{old}}_{ji} \; , \tag{131}$$

where $q^{\uparrow} = D^{\uparrow,\mathrm{new}}/D^{\uparrow,\mathrm{old}}$.

10.18 Evaluating the non-local pseudopotential energy

The action of the non-local pseudopotential on the wave function can be written as a sum of contributions from each electron and each angular momentum channel. The contribution to the local energy made by the non-local pseudopotential is

$$\begin{aligned} V_{\rm nl} &= \Psi^{-1}\hat{V}_{\rm nl}\Psi \\ &= \sum_i \Psi^{-1}\hat{V}^{\rm ps}_{{\rm nl},i}\Psi = \sum_i V_{{\rm nl},i} \ , \end{aligned} \tag{132}$$

where for simplicity we consider the case of a single atom placed at the origin. $V_{{\rm nl},i}$ may be written as [19]

$$\begin{aligned} V_{{\rm nl},i} &= \sum_l V^{\rm ps}_{{\rm nl},l}(r_i)\,\frac{2l+1}{4\pi}\int P_l\left[\cos(\theta'_i)\right] \\ &\quad\times \frac{\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_{i-1},\mathbf{r}'_i,\mathbf{r}_{i+1},\ldots,\mathbf{r}_N)}{\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_{i-1},\mathbf{r}_i,\mathbf{r}_{i+1},\ldots,\mathbf{r}_N)}\,d\Omega_{\mathbf{r}'_i} \ , \end{aligned} \tag{133}$$

where P_l denotes a Legendre polynomial.

CASINO currently performs the non-local projections for $l = 0, 1, 2$ only. The integral over the surface of the sphere in Eq. 133 is evaluated numerically. The $\mathbf{r}'$ dependence of the many-body wave function is expected to have predominantly the angular momentum character of the orbitals in the Slater part of the wave function. A suitable integration scheme is therefore to use a quadrature rule that integrates products of spherical harmonics exactly up to some maximum value $l_{\rm max}$. The quadrature grids currently available are listed in Table 1. To avoid bias the orientation of the axes is chosen randomly each time such an integral is evaluated.

Table 1: Quadrature grids for the non-local integration. NLRULE is the label for the rule (see Section 7.1), $l_{\rm max}$ is the maximum value of l which is integrated exactly and N_p is the number of points in the grid.

NLRULE	$l_{\rm max}$	N_p
1	0	1
2	2	4
3	3	6
4	5	12
5	5	18
6	7	26
7	11	50

Within a VMC calculation it is often possible to use a low-order quadrature rule because the error cancels over the run, but higher accuracy is required for wave function optimization and DMC calculations, which are biased by errors in the non-local integration. In principle the non-local energy should be summed over all the ionic cores and all electrons in the system. However, since the non-local potential of each ion is short ranged, one need only sum over the few atoms nearest to each electron.

Exact sampling of the non-local energy with DMC is problematic and we use the *localization approximation* in which the non-local operator acts on the trial wave function in exactly the same way as in VMC. The error introduced by this approximation is proportional to $(\Psi - \Psi_0)^2$ [3], where Ψ_0 is the exact wave function.

10.19 The core polarization potential energy

Core polarization potentials (CPPs) account for the polarization of the pseudo-ion cores by the fields of the other charged particles in the system. The polarization of the pseudo-ion cores by the fields of the valence electrons is a many-body effect which includes some of the core-valence correlation energy.[32] In the CPP approximation the polarization of a particular core is determined by the electric field at

the nucleus. The electric field acting on a given ion core at $\mathbf{R}_I$ due to the other ion cores at $\mathbf{R}_J$ and the electrons at $\mathbf{r}_i$ is

$$\mathbf{F}_I = -\sum_{J\neq I} Z_J \frac{\mathbf{R}_{JI}}{|\mathbf{R}_{JI}|^3} + \sum_i \frac{\mathbf{r}_{iI}}{|\mathbf{r}_{iI}|^3} \ , \tag{134}$$

where $\mathbf{R}_{JI} = \mathbf{R}_J - \mathbf{R}_I$ and $\mathbf{r}_{iI} = \mathbf{r}_i - \mathbf{R}_I$. The CPP energy is then

$$V_{\rm CPP} = -\frac{1}{2}\sum_I \alpha_I \mathbf{F}_I \cdot \mathbf{F}_I \ , \tag{135}$$

where α_I is the dipole polarizability of core I.

Eq. 134 assumes a classical description which is valid when the valence electrons are far from the core. When a valence electron penetrates the core the classical result is a very poor approximation, diverging at the nucleus. To remove this unphysical behaviour each contribution to the electric field in Eq. 134 is multiplied by a cutoff function $f(r_{iI}/\bar{r}_I)$, which tends to unity at large r_{iI}. A further possible modification is to allow the one-electron term in Eq. 135, which takes the form $-\alpha_I/(2r_{iI}^4)$, to depend on the angular momentum component, l, so that $\bar{r}_I$ in the cutoff function is replaced by $\bar{r}_{lI}$. With these modifications the CPP energy operator becomes

$$V_{\rm CPP} = -\frac{1}{2}\sum_I \alpha_I \left[\sum_i \frac{1}{r_{iI}^4} \sum_l f\left(\frac{r_{iI}}{\bar{r}_{lI}}\right)^2 \hat{P}_l + \sum_i \sum_{j\neq i} \frac{\mathbf{r}_{iI}\cdot\mathbf{r}_{jI}}{r_{iI}^3 r_{jI}^3} f\left(\frac{r_{iI}}{\bar{r}_I}\right) f\left(\frac{r_{jI}}{\bar{r}_I}\right) \right.$$
$$\left. -2\sum_i \sum_{J\neq I} \frac{\mathbf{r}_{iI}\cdot\mathbf{R}_{JI}}{r_{iI}^3 R_{JI}^3} f\left(\frac{r_{iI}}{\bar{r}_I}\right) Z_J + \left(\sum_{J\neq I} \frac{\mathbf{R}_{JI}}{R_{JI}^3} Z_J\right)^2 \right] \ , \tag{136}$$

where $\hat{P}_l$ is the projector onto the lth angular momentum component of the ith electron with respect to the Ith ion.

We use the cutoff function [31, 32],

$$f\left(x\right) = \left(1 - e^{-x^2}\right)^2 \ . \tag{137}$$

For efficient evaluation, Eq. 136 is written as

$$V_{\rm CPP} = -\frac{1}{2}\sum_I \alpha_I |\bar{\mathbf{F}}_I|^2 + \frac{1}{2}\sum_I \alpha_I \sum_i \frac{1}{r_{iI}^4} \sum_{l=0}^{lmx} \left[f\left(\frac{r_{iI}}{\bar{r}_I}\right)^2 - f\left(\frac{r_{iI}}{\bar{r}_{lI}}\right)^2 \right] \hat{P}_l \ , \tag{138}$$

where

$$\bar{\mathbf{F}}_I = -\sum_{J\neq I} Z_J \frac{\mathbf{R}_{JI}}{|\mathbf{R}_{JI}|^3} + \sum_i \frac{\mathbf{r}_{iI}}{|\mathbf{r}_{iI}|^3} f\left(\frac{r_{iI}}{\bar{r}_I}\right) \ , \tag{139}$$

and the maximum angular momentum is $lmx = 2$. In our approach the cutoff parameter for all angular momenta $l > 2$ is $\bar{r}_I$, which is slightly different from Shirley and Martin [32] who use $\bar{r}_{2I}$.

10.19.1 Implementation of CPPs in molecules and solids

Eq. 138 contains 5 parameters for each ion, $\alpha_I, \bar{r}_{0I}, \bar{r}_{1I}, \bar{r}_{2I}$ and $\bar{r}_I$, whose values are entered at the end of the xx_pp.data file, see Section 7.3. Suitable values of the parameters are given in the paper by Shirley and Martin.[32] If $\bar{r}_{0I} = \bar{r}_{1I} = \bar{r}_{2I} = \bar{r}_I$, the second term in Eq. 138 is zero and it is not calculated. The second term in Eq. 138 is short-ranged because $f(x) \to 1$ at large x. This term is calculated in real space.

The second term in Eq. 138 is added to the pseudopotential and the core radii LCUTOFFTOL and NLCUTOFFTOL are determined from the resulting potential. The electric field evaluation is

activated by the presence of core polarization terms in the pseudopotential files; they are not calculated by default since they may be expensive, especially when periodic boundary conditions are used.

In periodic boundary conditions the electric fields are evaluated directly from the analytic first derivatives of the Ewald potential, see section 10.20. Calculations using CPPs may be 5-10 percent slower than ones without CPPs in periodic systems.

NB: the first derivatives of the periodic potential in 1D polymers has not yet been implemented, and thus the core-polarization energy cannot be evaluated in such systems.

10.20 Evaluation of infinite Coulomb sums

10.20.1 3D Ewald interaction

In three dimensionally periodic systems, the periodic potential of a neutralized lattice of point charges may be evaluated using the Ewald method [1, 2]. Consider the periodic charge density consisting of a unit point charge at $\mathbf{r}_j$ in every simulation cell plus a uniform cancelling background,

$$\rho_j(\mathbf{r}) = \sum_{\mathbf{R}} \left(\delta(\mathbf{r} - \mathbf{r}_j - \mathbf{R}) - \frac{1}{\Omega} \right) , \tag{140}$$

where $\mathbf{R}$ denotes the lattice translation vectors and Ω is the volume of the simulation cell. The Ewald formula for the periodic potential corresponding to this charge density is

$$\begin{aligned} v_{\mathrm{E}}(\mathbf{r}, \mathbf{r}_j) &= \sum_{\mathbf{R}} \frac{\mathrm{erfc}\left(\gamma^{\frac{1}{2}} |\mathbf{r} - (\mathbf{r}_j + \mathbf{R})|\right)}{|\mathbf{r} - (\mathbf{r}_j + \mathbf{R})|} - \frac{\pi}{\Omega\gamma} \\ &+ \frac{4\pi}{\Omega} \sum_{\mathbf{G} \neq \mathbf{0}} \frac{\exp\left(-G^2/4\gamma\right)}{G^2} \exp(i\mathbf{G} \cdot (\mathbf{r} - \mathbf{r}_j)) , \end{aligned} \tag{141}$$

where $\mathbf{G}$ denotes the reciprocal lattice translation vectors. The value of $v_{\mathrm{E}}(\mathbf{r}, \mathbf{r}_j)$ is in principle independent of the screening parameter γ and also in practice providing enough vectors are included in the sums.[14] The larger the value of γ the more rapidly convergent is the real space sum, but the more slowly convergent is the reciprocal space sum. A compromise is required to minimize the overall computational cost. In CASINO, this parameter is set to $(2.8/\Omega^{1/3})^2$ which approximately minimizes the cost for a wide variety of Bravais lattices [37].

The full periodic potential of the simulation cell is obtained by adding the potentials of all the N charges and their cancelling backgrounds (which sum to zero because the cell has no net charge),

$$v(\mathbf{r}) = \sum_{j=1}^{N} q_j v_{\mathrm{E}}(\mathbf{r}, \mathbf{r}_j) . \tag{142}$$

The potential acting on the charge at $\mathbf{r}_i$ is therefore

$$v(\mathbf{r}_i) = \sum_{j(\neq i)}^{N} q_j v_{\mathrm{E}}(\mathbf{r}_i, \mathbf{r}_j) + q_i \xi , \tag{143}$$

where

$$\begin{aligned} \xi &= \lim_{\mathbf{r} \to \mathbf{r}_i} \left(v_{\mathrm{E}}(\mathbf{r}, \mathbf{r}_i) - \frac{1}{|\mathbf{r} - \mathbf{r}_i|} \right) \\ &= \sum_{\mathbf{R} \neq \mathbf{0}} \frac{\mathrm{erfc}(\gamma^{\frac{1}{2}} R)}{R} - \frac{2\gamma^{\frac{1}{2}}}{\pi^{\frac{1}{2}}} - \frac{\pi}{\Omega\gamma} \end{aligned} \tag{144}$$

[14] Increasing the input parameter EWALD_CONTROL will increase the number of reciprocal vectors included in the sum, the effect of which is to increase the range of γ over which the energy is constant. The default value of γ should normally lie in the middle of the constant energy region for the default number of reciprocal vectors, so playing with EWALD_CONTROL is normally unnecessary.

$$+\frac{4\pi}{\Omega}\sum_{\mathbf{G}\neq\mathbf{0}}\frac{\exp(-G^2/4\gamma)}{G^2} \tag{145}$$

is the potential acting on the charge at $\mathbf{r}_i$ due to its own images and cancelling background. The full Ewald potential energy appearing in the QMC Hamiltonian is therefore

$$\begin{aligned} U(\mathbf{r}_1,\ldots,\mathbf{r}_N) &= \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\(j\neq i)}}^{N} q_i q_j v_{\mathrm{E}}(\mathbf{r}_i,\mathbf{r}_j)+\frac{\xi}{2}\sum_{i=1}^{N} q_i^2 & (146)\\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\(j\neq i)}}^{N} q_i q_j \left(v_{\mathrm{E}}(\mathbf{r}_i,\mathbf{r}_j)-\xi\right)\ , & (147)\end{aligned}$$

where we have used the charge neutrality condition, $q_i=-\sum_{j(\neq i)} q_j$. The interaction $v_{\mathrm{E}}(\mathbf{r}_i,\mathbf{r}_j)-\xi$ approaches $1/|\mathbf{r}_i-\mathbf{r}_j|$ as $\mathbf{r}_i\rightarrow\mathbf{r}_j$ and is independent of the choice of the zero of potential.

The gradient of the 3D Ewald potential (Eq. 141) is required for evaluation of the core polarization contribution to the total energy (Section 10.19). It is given by

$$\begin{aligned} \nabla v_{\mathrm{E}}(\mathbf{r},\mathbf{r}_j) &= -\sum_{\mathbf{R}}\frac{\mathbf{r}-(\mathbf{r}_j+\mathbf{R})}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|^2}\left(\frac{\mathrm{erfc}\left(\gamma^{\frac{1}{2}}|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|\right)}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|}+\frac{2\gamma^{\frac{1}{2}}}{\pi^{\frac{1}{2}}}\exp(-\gamma|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|^2)\right)\\ &- \frac{4\pi}{\Omega}\sum_{\mathbf{G}\neq\mathbf{0}}\mathbf{G}\frac{\exp\left(-G^2/4\gamma\right)}{G^2}\sin\left(\mathbf{G}\cdot(\mathbf{r}-\mathbf{r}_j)\right)\ . \end{aligned} \tag{148}$$

10.20.2 2D Ewald interaction

Infinite Coulomb sums in systems which are periodic in two dimensions (the xy-plane, according to CASINO) are performed using the standard 2D Ewald method originally developed by Parry [38]. One way to derive the relevant formula is to take the infinite separation limit of the 3D sum for a periodic stack of finite-width slabs. CASINO uses this algorithm when treating two-dimensional slabs of atoms with local Gaussian basis sets (useful in modelling surfaces) and also when treating 2D electron and electron-hole phases (either as strict 2D planes, 2D slabs with finite thickness, or strict 2D bilayers). In the case of periodic arrays of slabs separated by a finite vacuum gap (necessary when using plane-wave basis sets) then the regular 3D algorithm is used.

Consider a finite width slab with a charge density periodic in two dimensions consisting of a unit point charge at $\mathbf{r}_j$ in every simulation cell plus a uniform cancelling background.

$$\rho_j(\mathbf{r})=\sum_{\mathbf{R}}\left(\delta(\mathbf{r}-\mathbf{r}_j-\mathbf{R})-\frac{1}{A}\right)\ , \tag{149}$$

where $\mathbf{R}$ now denotes the 2D lattice translation vectors in the xy-plane, and A is the area of the simulation cell in that plane. The Parry formula for the periodic potential corresponding to this charge density is

$$\begin{aligned} v_{\mathrm{E}}^{2D}(\mathbf{r},\mathbf{r}_j) &= \sum_{\mathbf{R}}\frac{\mathrm{erfc}\left(\gamma^{\frac{1}{2}}|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|\right)}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|}\\ &+ \frac{\pi}{A}\sum_{\mathbf{G}\neq\mathbf{0}}\frac{\exp(i\mathbf{G}\cdot(\mathbf{r}-\mathbf{r}_j))}{G}\left[\exp(zG)\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}+z\gamma^{\frac{1}{2}}\right)+\exp(-zG)\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}-z\gamma^{\frac{1}{2}}\right)\right]\\ &- \frac{\pi}{A}\left[\mathrm{erf}(z\gamma^{\frac{1}{2}})z+\frac{\exp(-\gamma z^2)}{(\gamma\pi)^{\frac{1}{2}}}\right]\ , \end{aligned} \tag{150}$$

where $\mathbf{G}$ denotes the reciprocal lattice translation vectors in the xy-plane, and z is the z-component of the $\mathbf{r}-\mathbf{r}_j$ vector. In two dimensions CASINO sets the screening parameter γ to $(2.4/A^{\frac{1}{2}})^2$ which again should approximately minimize the cost over different Bravais lattices.

The full periodic potential of the simulation cell is obtained by following a procedure analogous to that described for the 3D case, with the self term ξ given by

$$\xi = \lim_{\mathbf{r}\to\mathbf{r}_i}\left(v_{\mathrm{E}}(\mathbf{r},\mathbf{r}_i)-\frac{1}{|\mathbf{r}-\mathbf{r}_i|}\right) \tag{151}$$

$$= \sum_{\mathbf{R}\neq\mathbf{0}}\frac{\mathrm{erfc}\left(\gamma^{\frac{1}{2}}R\right)}{R}-\frac{2\gamma^{\frac{1}{2}}}{\pi^{\frac{1}{2}}}+\frac{\pi}{2A}\sum_{\mathbf{G}\neq\mathbf{0}}\frac{\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}\right)}{G}-\frac{\pi^{\frac{1}{2}}}{A\gamma^{\frac{1}{2}}}\,. \tag{152}$$

The first derivatives of the 2D Ewald potential, required for the evaluation of the core polarization energy in 2D slabs, are different in directions parallel and perpendicular to the plane of the slab. The x and y derivatives are given by

$$\frac{\partial v_{\mathrm{E}}^{2D}(\mathbf{r},\mathbf{r}_j)}{\partial\lambda} = -\sum_{\mathbf{R}}\frac{(\mathbf{r}-(\mathbf{r}_j+\mathbf{R}))_\mu}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|^2}\left[\frac{\mathrm{erfc}\left(\gamma^{\frac{1}{2}}|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|\right)}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|}+\frac{2\gamma^{\frac{1}{2}}}{\pi^{\frac{1}{2}}}\exp(-\gamma|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|^2)\right] \tag{153}$$

$$-\frac{\pi}{A}\sum_{\mathbf{G}\neq\mathbf{0}}\frac{G_\mu\sin(\mathbf{G}\cdot(\mathbf{r}-\mathbf{r}_j))}{G}\left[\exp(zG)\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}+z\gamma^{\frac{1}{2}}\right)+\exp(-zG)\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}-z\gamma^{\frac{1}{2}}\right)\right]\,.$$

where $\lambda = x$ or y, and the z derivative is given by

$$\frac{\partial v_{\mathrm{E}}^{2D}(\mathbf{r},\mathbf{r}_j)}{\partial z} = -\sum_{\mathbf{R}}\frac{z}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|^2}\left[\frac{\mathrm{erfc}\left(\gamma^{\frac{1}{2}}|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|\right)}{|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|}+\frac{2\gamma^{\frac{1}{2}}}{\pi^{\frac{1}{2}}}\exp(-\gamma|\mathbf{r}-(\mathbf{r}_j+\mathbf{R})|^2)\right]$$

$$+\frac{\pi}{A}\sum_{\mathbf{G}\neq\mathbf{0}}\cos(\mathbf{G}\cdot(\mathbf{r}-\mathbf{r}_j))\left[\exp(zG)\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}+z\gamma^{\frac{1}{2}}\right)+\exp(-zG)\mathrm{erfc}\left(\frac{G}{2\gamma^{\frac{1}{2}}}-z\gamma^{\frac{1}{2}}\right)\right]$$

$$-\frac{2\pi}{A}\mathrm{erf}(z\gamma^{\frac{1}{2}})\,. \tag{154}$$

Note finally that in things such as 2D bilayer systems (electrons in one layer, holes in the other, say) there is an additional 'capacitor term' due to interaction of the backgrounds, about which I shall write more here in a minute. CASINO does not evaluate this term.

10.20.3 1D Coulomb interaction

Coulomb sums in systems which are periodic in one dimension (the x-direction, according to CASINO) are performed using an algorithm based on the Euler-Maclaurin summation formula.

See for example Eq. 4.8 in Comp. Phys. Commun. **84**, 156 (1994).

More details to appear here later.

10.20.4 Model Periodic Coulomb Interaction

The Model Periodic Coulomb (MPC) interaction [10, 11, 12] is used to reduce finite size effects in periodic calculations. The exact MPC interaction operator is

$$\hat{H}_{e-e}^{\mathrm{exact}} = \sum_{i>j}f(\mathbf{r}_i-\mathbf{r}_j)+\sum_i\int_{\mathrm{WS}}\rho(\mathbf{r})\left[v_{\mathrm{E}}(\mathbf{r}_i-\mathbf{r})-f(\mathbf{r}_i-\mathbf{r})\right]d\mathbf{r}\,, \tag{155}$$

where $f(\mathbf{r})$ is the $1/r$ Coulomb interaction treated within the "nearest image" convention, which corresponds to reducing the vector $\mathbf{r}$ into the Wigner-Seitz (WS) cell of the simulation cell, v_{E} is the Ewald potential, and ρ is the electronic charge density from the many-electron wave function Ψ. The

electron-electron contribution to the total energy is evaluated as the expectation value of $\hat{H}_{e-e}^{\rm exact}$ with Ψ minus a double counting term,

$$E_{\rm e-e}^{\rm exact} = \langle\Psi|\hat{H}_{e-e}^{\rm exact}|\Psi\rangle - \frac{1}{2}\int_{\rm WS}\rho(\mathbf{r})\rho(\mathbf{r}')\left[v_{\rm E}(\mathbf{r}-\mathbf{r}') - f(\mathbf{r}-\mathbf{r}')\right]\,d\mathbf{r}\,d\mathbf{r}'\ . \tag{156}$$

Evaluating the expectation value gives

$$\begin{aligned} E_{\rm e-e}^{\rm exact} &= \frac{1}{2}\int_{\rm WS}\rho(\mathbf{r})\rho(\mathbf{r}')v_{\rm E}(\mathbf{r}-\mathbf{r}')\,d\mathbf{r}\,d\mathbf{r}' \\ &+ \left(\int_{\rm WS}|\Psi|^2\sum_{i>j}f(\mathbf{r}_i-\mathbf{r}_j)\,\Pi_k\,{\rm d}\mathbf{r}_k - \frac{1}{2}\int_{\rm WS}\rho(\mathbf{r})\rho(\mathbf{r}')f(\mathbf{r}-\mathbf{r}')\,d\mathbf{r}\,d\mathbf{r}'\right)\ , \end{aligned} \tag{157}$$

where the first term on the right hand side is the Hartree energy and the term in brackets is the exchange-correlation energy. We can see that the Hartree energy is calculated with the Ewald interaction while the exchange-correlation energy (expressed as the difference between a full Coulomb term and a Hartree term) is calculated with the cutoff interaction f.

In a DMC calculation we require the local energy at every step, but we only know the DMC charge density, ρ, at the end of the run. Normally we have a good approximation to the charge density, $\rho_{\rm A}$, either from an independent particle calculation or a VMC calculation. We can avoid the need to know ρ exactly by constructing a new interaction operator which involves only $\rho_{\rm A}$,

$$\begin{aligned} \hat{H}_{e-e} &= \sum_{i>j}f(\mathbf{r}_i-\mathbf{r}_j) + \sum_i\int_{\rm WS}\rho_{\rm A}(\mathbf{r})\left[v_{\rm E}(\mathbf{r}_i-\mathbf{r}) - f(\mathbf{r}_i-\mathbf{r})\right]\,d\mathbf{r} \\ &- \frac{1}{2}\int\rho_{\rm A}(\mathbf{r})\rho_{\rm A}(\mathbf{r}')\left[v_{\rm E}(\mathbf{r}-\mathbf{r}') - f(\mathbf{r}-\mathbf{r}')\right]\,d\mathbf{r}\,d\mathbf{r}'\ . \end{aligned} \tag{158}$$

Because of the presence of the third term on the right hand side, which is a constant, there is no double counting and the interaction energy becomes

$$\begin{aligned} E_{\rm e-e} &= \langle\Psi|\hat{H}_{e-e}|\Psi\rangle \\ &= \int_{\rm WS}\rho(\mathbf{r})\rho_{\rm A}(\mathbf{r}')v_{\rm E}(\mathbf{r}-\mathbf{r}')\,d\mathbf{r}\,d\mathbf{r}' - \frac{1}{2}\int_{\rm WS}\rho_{\rm A}(\mathbf{r})\rho_{\rm A}(\mathbf{r}')v_{\rm E}(\mathbf{r}-\mathbf{r}')\,d\mathbf{r}\,d\mathbf{r}' \\ &+ \left(\int_{\rm WS}|\Psi|^2\sum_{i>j}f(\mathbf{r}_i-\mathbf{r}_j)\,\Pi_k\,d\mathbf{r}_k - \int_{\rm WS}\rho(\mathbf{r})\rho_{\rm A}(\mathbf{r}')f(\mathbf{r}-\mathbf{r}')\,d\mathbf{r}\,d\mathbf{r}'\right. \\ &+ \left.\frac{1}{2}\int_{\rm WS}\rho_{\rm A}(\mathbf{r})\rho_{\rm A}(\mathbf{r}')f(\mathbf{r}-\mathbf{r}')\,d\mathbf{r}\,d\mathbf{r}'\right)\ . \end{aligned} \tag{159}$$

Noting that

$$E_{\rm e-e} = E_{\rm e-e}^{\rm exact} - \frac{1}{2}\int_{\rm WS}\left[\rho(\mathbf{r})-\rho_{\rm A}(\mathbf{r})\right]\left[\rho(\mathbf{r}')-\rho_{\rm A}(\mathbf{r}')\right]\left[v_{\rm E}(\mathbf{r}-\mathbf{r}') - f(\mathbf{r}-\mathbf{r}')\right]\,d\mathbf{r}\,d\mathbf{r}'\ , \tag{160}$$

we see that the error due to this approximation is second order in $(\rho-\rho_{\rm A})$, and in addition the operator $(v_{\rm E}-f)$ becomes very small as the size of the simulation cell goes to infinity. The error term is therefore usually small and is neglected although it could be calculated after the simulation. We use the MPC expressions of Eqs. 158 and 159 in both VMC and DMC calculations.

The first term of the Hamiltonian of Eq. 158 is evaluated in real space and the second term in Fourier space. The third term is a constant which is evaluated in reciprocal space at the start of the calculation. Introducing the Fourier transformed quantities,

$$f_{\mathbf{G}} = \frac{1}{\Omega}\int_{\rm WS}f(\mathbf{r})e^{i\mathbf{G}\cdot\mathbf{r}}d\mathbf{r}\ , \tag{161}$$

$$\rho_{\mathbf{G}} = \frac{1}{\Omega}\int_{\rm WS}\rho(\mathbf{r})e^{i\mathbf{G}\cdot\mathbf{r}}d\mathbf{r}\ , \tag{162}$$

where Ω is the volume of the cell, and noting that the Fourier transform of the Ewald interaction is $\frac{4\pi}{\Omega G^2}$, we have

$$\hat{H}_{e-e} = \sum_{i>j} f(\mathbf{r}_i - \mathbf{r}_j) + \Omega \sum_i \sum_{\mathbf{G}\neq 0} \left[\frac{4\pi}{\Omega G^2} - f_{\mathbf{G}} \right] \rho_{A,\mathbf{G}} e^{-i\mathbf{G}\cdot\mathbf{r}_i} - \Omega \sum_i f_{\mathbf{G}=0} \rho_{A,\mathbf{G}=0} - C \; , \tag{163}$$

where

$$C = \frac{\Omega^2}{2} \sum_{\mathbf{G}\neq 0} \left[\frac{4\pi}{\Omega G^2} - f_{\mathbf{G}} \right] \rho^*_{A,\mathbf{G}} \rho_{A,\mathbf{G}} - \frac{\Omega^2}{2} f_{\mathbf{G}=0} \rho^*_{A,\mathbf{G}=0} \rho_{A,\mathbf{G}=0} \tag{164}$$

The calculation of $f_{\mathbf{G}}$ is achieved using the following scheme developed by Randy Hood. The integrand in Eq. 161 diverges at the origin and we separate out the divergent behaviour by writing

$$f(\mathbf{r}) = g(\mathbf{r}) + h(\mathbf{r}) \tag{165}$$

where

$$\begin{aligned} g(\mathbf{r}) &= y(r) && r < L \\ &= \frac{1}{r} && r > L \; , \end{aligned} \tag{166}$$

and

$$\begin{aligned} h(\mathbf{r}) &= \left(\frac{1}{r} - y(r) \right) && r < L \\ &= 0 && r > L \; , \end{aligned} \tag{167}$$

where L is the radius of the largest sphere which is contained within the Wigner-Seitz cell and $y(r)$ is chosen to be

$$y(r) = -\frac{r^2}{2L^3} + \frac{3}{2L} \; , \tag{168}$$

so that both g and h have continuous first derivatives at $r = L$. The Fourier transform of $h(\mathbf{r})$ is calculated analytically as

$$\begin{aligned} h_{\mathbf{G}} &= \frac{1}{\Omega} \int_0^L \int_{-1}^{+1} \left(\frac{1}{r} + \frac{r^2}{2L^3} - \frac{3}{2L} \right) 2\pi r^2 e^{iGr\cos\theta} d\cos\theta \, dr \\ &= \frac{4\pi}{\Omega G^2} + \frac{12\pi}{\Omega L^2 G^4} \left[\cos GL - \frac{\sin GL}{GL} \right] \; , \end{aligned} \tag{169}$$

from which the $G = 0$ value can be extracted as

$$h_{\mathbf{G}=0} = \frac{2\pi L^2}{5\Omega} \; , \tag{170}$$

The Fourier transform of g is calculated numerically with an FFT, added to the analytic Fourier transform of h, and stored by CASINO in the file eepot.data. Note that for historical reasons the quantity stored in eepot.data is actually $\Omega f_{\mathbf{G}}$.

10.21 Estimating equilibration times and correlation periods

The rms distance diffused by a particle in a period T of imaginary time is $\sqrt{2N_D DAT}$, where A is the move acceptance ratio (which is usually close to 1 in DMC and 1/2 in VMC), N_D is the dimensionality of the system (which is usually 3, unless a strict 2D system is being studied) and $D = 1/2m$ is the diffusion constant, where m is the particle mass (NB, $D = 1/2$ for electrons). We expect that correlation effects will disappear when the particles have diffused through distances in excess of the physically-relevant length-scale λ. (For example, in an electron gas, the density parameter r_s is the relevant length-scale.) Let $T = N_{\mathrm{move}} \times \tau$, where N_{move} is the number of moves and τ is the timestep. Then the number of moves over which we expect correlation effects to be present is

$$N_{\mathrm{move}} = \frac{\lambda^2}{2N_D D\tau A}. \tag{171}$$

The number of equilibration moves should be substantially larger than the above estimate of the correlation period in order to ensure that all of the transient effects due to the initial distribution die away. The required equilibration period is often considerably greater than one might expect by simply examining the variation of the total energy with time.

In practical QMC calculations, with sensible choices of timestep, we often find the VMC correlation period to be about 5 configuration moves and the DMC correlation period to be about 1000 moves.

10.22 Statistical analysis of data

The Monte Carlo data must be analysed to obtain the required mean values along with a measure of their statistical errors. Two factors complicate the analysis: (1) the data is normally serially correlated, and (2) there is an initial period of equilibration which should be excluded from the averaging.

We use the "blocking" method [9] in which adjacent data points are averaged to form block averages:

$$x_i^1 = \frac{1}{2}(x_{2i-1} + x_{2i}) \; . \tag{172}$$

This procedure is carried out recursively so that after each blocking transformation the number of data points is reduced by one half. For simplicity we assume that the number of data points, M, is a power of two. The mean value of the block means is unchanged by the block transformations, but the variance of the block means tends to increase with the number of block transformations,

$$\sigma_b^2 = \frac{1}{M_b - 1} \sum_{j=1}^{M_b} (\overline{x_{bj}^2} - \bar{x}^2) \; , \tag{173}$$

where M_b is the number of blocks, $\bar{x}_{bj}$ is the mean value of the jth block and $\bar{x}$ is the overall mean value. The error in σ_b^2 is estimated from

$$\sqrt{\frac{2}{M/M_b - 1}}\, \sigma_b^2 \; . \tag{174}$$

The value of σ_b increases with block length M_b until a limiting value is reached which corresponds to the true standard deviation of the mean. For very large block sizes the error in σ_b becomes large and if M is not large enough a limiting value of σ_b is not reached, but because it is simple to calculate the error in σ_b, such behaviour is easy to identify. These operations are carried out by the utility "reblock", which reads data from a "dmc.hist" or "vmc.hist" file.

10.22.1 Estimate of the correlation time given by CASINO

The correlation time of the energy is computed and shown for every block in a VMC calculation, and also when using the reblock utility. The correlation time measures the average number of Monte Carlo steps between two uncorrelated values of the energy, and should be unity for optimal statistics. Note that by "Monte Carlo Step" what is meant is "every step for which an energy is stored", in this context. For example, in a VMC calculation, every CORPER×NVMCAVE steps are merged into one energy. Increasing any of the two variables would decrease the calculated correlation time, meaning that the values that have been stored are more uncorrelated.

The definition of the correlation time τ of an observable H is:

$$\tau = \int_{-\infty}^{+\infty} A(t)dt = \int_{-\infty}^{+\infty} \frac{\langle (H_{t'} - \langle H_{t''} \rangle_{t''})(H_{t'+t} - \langle H_{t''} \rangle_{t''}) \rangle_{t'}}{\sigma_H^2} dt \tag{175}$$

where $A(t)$ is the value of the autocorrelation function at an interval of t, $\sigma_H^2 = \langle (H_{t'} - \langle H_{t''} \rangle_{t''})^2 \rangle_{t'}$ is the variance of the expectation values, and the latter are taken with respect to their subscript, which

we shall remove for the sake of clarity. For a discrete set of values, equally spaced by an amount $\Delta t = 1$:

$$\tau = \sum_{t=-\infty}^{+\infty} A(t) = 1 + 2\sum_{t=1}^{+\infty} A(t) = 1 + 2\sum_{t=1}^{+\infty} \frac{\langle (H_{t'} - \langle H \rangle)(H_{t'+t} - \langle H \rangle) \rangle}{\sigma_H^2} \tag{176}$$

and for a finite set of length N:

$$\tau = 1 + 2\sum_{t=1}^{N-1} \frac{\langle (H_{t'} - \langle H \rangle)(H_{t'+t} - \langle H \rangle) \rangle}{\sigma_H^2} \tag{177}$$

Numerically, the problem with this expression is that if averages are used instead of proper expectation values (which are, of course, unknown), great fluctuations will appear at the tail of the autocorrelation function. This problem is solved by introducing a cut-off in the summation:

$$\tau(t_{max}) = 1 + 2\sum_{t=1}^{t_{max}} \frac{\overline{(H_{t'} - \overline{H})(H_{t'+t} - \overline{H})}}{\hat{\sigma}_H^2} \tag{178}$$

where the numerator is the average of the measured values of its arguments over the configurations indexed by $1 \leq t' \leq N - t$, and $\hat{\sigma}_H^2$ is the variance of these measures. One possibility for setting the cut-off is to check against the self-consistent inequality $t_{max} < 3\tau(t_{max})$ while computing the sum, and truncate it as soon as it stops being true. This allows for an estimate of the error in above expression to be calculated:

$$\epsilon_\tau(t_{max}) = \tau\sqrt{\frac{2(2t_{max} + 1)}{N}} \tag{179}$$

10.22.2 Further considerations for DMC

Each iteration is equally weighted in a VMC calculation; however, for DMC, each iteration is weighted by the total weight of the configurations multiplied by the Π-weight. In either case, let the iteration weights be w_i, the total number of data points be M and the energy from iteration i be e_i.

Consider the bth reblocking transformation, in which the block length is $B_b = 2^{b-1}$. The data range may be divided into M_b blocks, the last of which is usually incomplete.

For each block j, the block weight is

$$W_{bj} = \sum_{i \in j} w_i, \tag{180}$$

and the corresponding block energy is

$$E_{bj} = \frac{\sum_{i \in j} e_i w_i}{W_{bj}}. \tag{181}$$

The "reblocked" energy is

$$\begin{aligned} E_b &= \frac{\sum_j E_{bj} W_{bj}}{\sum_j W_{bj}} \\ &= \frac{\sum_i e_i w_i}{\sum_i w_i} \equiv E, \end{aligned} \tag{182}$$

which is therefore independent of reblocking transformation. On the other hand, the reblocked variance is

$$\sigma_b^2 = \frac{\sum_j W_{bj}(E_{bj} - E)^2}{\sum_j W_{bj} - \frac{\sum_j W_{bj}^2}{\sum_j W_j}}, \tag{183}$$

which *does* depend on the reblocking transformation number.

The number of blocks at the bth reblocking transformation is $N_b = M/B_b$. (Note that N_b is not necessarily an integer, because the last block may be incomplete.) The standard error in the energy estimate at reblocking transformation number b is

$$\delta_b = \frac{\sigma_b}{\sqrt{N_b}}, \tag{184}$$

and the error in δ_b may be estimated as

$$\epsilon_b = \frac{\delta_b}{\sqrt{2(N_b - 1)}}. \tag{185}$$

The reblock utility produces a table of δ_b and ϵ_b against b; the user can then look for a region in which the standard error δ_b has plateaued as a function of b, and choose an appropriate reblocking transformation number, which is then used to calculate the error bars on all the different components of energy.

10.23 Wave function optimization - standard method

Optimization of the trial wave function is a crucial part of a VMC or DMC calculation. CASINO allows optimization of the parameters in the Jastrow factor, see Section 7.2, of the coefficients of the determinants of a multi-determinant wave function, of the parameters in the backflow η-function, of pairing parameters in electron-hole gases and Padé coefficients in orbitals in Wigner crystals.

Wave function optimization in CASINO is achieved by minimizing the variance of the energy,

$$\sigma_E^2(\alpha) = \frac{\int \Psi^2(\alpha)\,[E_L(\alpha) - E_V(\alpha)]^2\,d\mathbf{R}}{\int \Psi^2(\alpha)\,d\mathbf{R}}\,, \tag{186}$$

where α is a set of parameters, and E_V is the variational energy. The most important ground for preferring variance minimization to, say, energy minimization is that it shows better numerical stability, particularly in large systems.

Minimization of σ_E^2 is carried out via a correlated-sampling approach in which a set of configurations distributed according to $\Psi^2(\alpha_0)$ is generated, where α_0 is an initial set of parameter values. The variance $\sigma_E^2(\alpha)$ is then evaluated as

$$\sigma_E^2(\alpha) = \frac{\int \Psi^2(\alpha_0)\,w(\alpha)\,[E_L(\alpha) - E_V(\alpha)]^2\,d\mathbf{R}}{\int \Psi^2(\alpha_0)\,w(\alpha)\,d\mathbf{R}}\,, \tag{187}$$

where

$$E_V(\alpha) = \frac{\int \Psi^2(\alpha_0)\,w(\alpha)\,E_L(\alpha)\,d\mathbf{R}}{\int \Psi^2(\alpha_0)\,w(\alpha)\,d\mathbf{R}}\,, \tag{188}$$

and the integrals contain a weighting factor, $w(\alpha)$, given by

$$w(\alpha) = \frac{\Psi^2(\alpha)}{\Psi^2(\alpha_0)}\,. \tag{189}$$

The parameters α are adjusted until $\sigma_E^2(\alpha)$ is minimized. The advantage of the correlated-sampling approach is that one does not have to generate a new set of configurations every time the parameter values are changed.

In order to generate a set of configurations distributed according to $\Psi^2(\alpha_0)$, a VMC 'configuration generation' run must be carried out first. The subsequent variance minimization using these configurations is handled by the subroutine VARMIN. The minimization itself is carried out by the routine NL2SNO in module NL2SOL, which performs an unconstrained minimization (without requiring derivatives) of a sum of m squares of functions which contain n variables, where $m \geq n$. (Information on the minimization routine can be found in Reference [39]). Having generated a new set of parameters, we then carry out a configuration generation run with these parameters, and, if necessary, a further variance minimization run, and so forth.

Before carrying out this process, the user must decide how they wish to parameterize the trial wave function, and with how many parameters. They must also decide on the number of configurations to

be used in the optimization. These choices are system specific, and depend on the level of accuracy to which the user wishes to work.

A systematic approach to deciding on an appropriate number of variational parameters is to start by optimizing a few parameters, then to add more and re-optimize, and so on, until the decrease in energy resulting from the inclusion of additional parameters is comparable with the error bars in the VMC energy. Once this stage has been reached, adding further parameters can be of no benefit; indeed the extra variational freedom will then become problematic, leading to poorer trial wave functions, as parameters are optimized for specific configuration sets. If further parameters are to be optimized, the user must increase the number of configurations.

It is clearly desirable for the VMC-generated configurations to be completely uncorrelated. This can be achieved by giving **corper** a large value. Reblocking VMC energies in a preliminary VMC run will allow the user to determine the correlation period for VMC energies, which in turn suggests a suitable value for **corper**. It is also essential that the VMC configuration generation run is fully equilibrated. Since VMC equilibration is usually computationally inexpensive, this should be straightforward enough. The utility *plot_vmc_energy* can be used to verify that the VMC energies have equilibrated.

The user may choose whether to optimize the Jastrow function, determinant expansion coefficients, or the pairing parameter or Padé coefficients in a Wigner crystal, by setting the **vm_opt_jasfun**, **vm_opt_detcoeff**, and **vm_opt_pairing** flags as appropriate. For most applications, it is only necessary to optimize the Jastrow function. Exactly which parameters in the Jastrow function are to be optimized are indicated in the `jastrow.data` file. (See Section 10.14.)

When optimizing a Jastrow function, if non-linear parameters such as the cutoffs are not being optimized, it is possible to accelerate the variance minimization process by setting **vm_mode** to 'linear'; CASINO will then exploit the fact that the Jastrow function has a linear dependence on all the parameters appearing in it, enormously speeding up the process of recomputing configuration energies when the parameters are changed. This is expensive in memory, however. If memory requirements prevent the use of linear mode, or cutoffs are to be included in the optimization, then **vm_mode** should be set to 'direct', and the local energies of each of the configurations will be calculated from scratch when the parameters are changed [NOTE : THIS STILL ONLY WORKS FOR THE OLD JASTROW FACTOR - TO BE CHANGED SOON. MDT 6.2004].

When optimizing Padé coefficients, the condition that the coefficients are the same for electrons of either spin can be enforced by setting the 'optimize all parameters' flag in `heg.data` to '.FALSE.'. The second pair of Padé coefficients in the file are then ignored. If Padé coefficients are to be optimized, 'direct' mode must be used.

The process of variance minimization requires careful monitoring by the user. Unless proper action is taken, the correlated sampling procedure may become numerically unstable, particularly for large system sizes. The characteristic of these instabilities is that during the minimization procedure a few configurations (often only one) acquire a very large weight, $w(\alpha)$. The estimate of the variance is then reduced almost to zero by a set of parameters which are found to give extremely poor results in a subsequent QMC calculation. One can overcome this instability by using more configurations. Various alternative ways of dealing with this instability have been devised. One method is to limit the upper value of the weights [15] or to set the weights equal to unity [17, 14]. In our calculations for large systems we normally set the weights to unity, which is achieved by setting **vm_reweight** to '.FALSE.'. It can be verified that the effect of this approximation is in general negligible by regenerating configurations and carrying out a new variance minimization process.

On the other hand, if the initial trial wave function is poor, then some of the configuration generation/variance minimization cycles can be bypassed by using the weights. There is also some evidence that the optimization of "difficult" parameters such as cutoff lengths works better when **vm_reweight** is '.TRUE.'.

It is possible to choose how much information CASINO will provide during the minimization process. Setting **vm_info** to 1 will provide no information during the minimization; setting it to 2 will provide a list of the parameters, the mean energy and variance at each iteration; 3 will provide the same information as 2, but the weights will also be computed and their mean and variance displayed (note that the weights are not actually used unless **vm_reweight** is set); 4 provides an enormous amount of detail and is only likely to be of use for development or debugging purposes.

Some things to check when carrying out variance minimization:

- Does the VMC energy of successive configuration generation runs decrease by a statistically significant amount? Once it has ceased to fall, there is little point in continuing the minimization process. Usually, for a successful run, we find that a single configuration generation / variance minimization loop is sufficient, but this should always be checked. If the energy fails to 'settle down' over successive loops then there may be too much variational freedom in the trial wave function.
- Does the reported variance at each iteration in the variance minimization runs fall? If not then there is a problem with the minimizer (NL2SOL) itself.
- Does the reported mean energy at each iteration fall? Since energy and variance minima do not necessarily coincide, it is possible for it to rise slightly; however, if it rises substantially then it suggests there may be a problem such as the use of too many parameters.
- Do the parameters themselves change values substantially in successive optimizations when the energy is not being lowered? If 'yes' then they may well be redundant.

10.24 Wave function optimization - new method

10.24.1 Background

Consider the linear parameters in CASINO's new Jastrow factor (the expansion coefficients $\{\alpha_l\}$, $\{\beta_m\}$, $\{\gamma_{lmn}\}$, $\{a_A\}$ and $\{b_B\}$ in the u, χ, f, p and q terms respectively). The local energy of a single configuration can be shown to be a quadratic function of the linear parameters; hence the variance of the local energies of a fixed, finite set of configurations is a quartic function of the parameters. But this is precisely the quantity that is minimized in an unreweighted variance-minimization calculation. The process of variance minimization can therefore be greatly simplified and accelerated if only linear Jastrow parameters are to be optimized.

In an ordinary variance-minimization calculation, the VMC method is used to generate a set of configurations distributed according to the initial trial wave function. During the optimization process, the variance of the local energies of this configuration set is computed for different sets of parameters, and the variance is minimized with respect to the parameters. By contrast, in the new optimization method, the quartic expansion coefficients of the unreweighted variance are accumulated directly in VMC: there is no need to write out a set of configurations. Furthermore, when the unreweighted variance—referred to as the *least-squares function* (LSF)—is evaluated during the subsequent optimization stage, there is no need to repeatedly sum over a set of configurations: the quartic LSF can be evaluated directly.

The fact that the LSF can be evaluated as a quartic function of the parameters gives two significant advantages over the standard variance-minimization algorithm: (i) the LSF can be evaluated extremely rapidly (typically thousands of times per second); furthermore the CPU time required is *independent* of the system size; and (ii) the LSF along any line in parameter space is a simple quartic polynomial, so that the exact, *global* minimum of the LSF along that line can be computed analytically.

The method has two drawbacks: (i) only linear Jastrow parameters can be optimized in this fashion; and (ii) the number of quartic coefficients to be evaluated and stored in memory grows as the fourth power of the number of parameters to be optimized.

10.24.2 Using the new optimization method

A CASINO variance-minimization calculation using the new optimization method is carried out in exactly the same way as an ordinary variance-minimization calculation except that:

1. The **use_newopt** keyword in `input` should be set to "T".
2. There is no need to write out any configurations, so the user can set the **nwrcon** keyword to zero.
3. If desired, the user may change the method used to minimize the LSF with respect to the set of parameters by using the **newopt_method** keyword. This can take the values: "CG" (*conjugate gradients*); "SD" (*steepest descents*); "GN" (*Gauss–Newton*); "MC" (*Monte Carlo*

line minimization); "LM" (*simple line minimization*); "CG_MC" (*alternate conjugate gradients and Monte Carlo line minimization*); "BFGS" (*Broyden–Fletcher–Goldfarb–Shanno*); or "BFGS_MC" (*alternate BFGS and Monte Carlo line minimization*). If the **newopt_method** keyword is omitted then the BFGS method will be used by default. If you experience difficulty optimizing a large set of parameters then the Gauss–Newton method is worth trying. The BFGS method seems to be the most efficient method in general, however.

4. If desired, the user can change both the maximum number of iterations and the number of line minimizations to be performed by means of the **newopt_iterations** keyword. If this keyword is omitted, or it is given a negative value, then a default number of iterations will be performed.

The cutoff lengths in the Jastrow factor are important variational parameters, and some attempt to optimize them should always be made. It is recommended that a (relatively cheap) calculation using the standard variance-minimization method should be carried out in order to optimize the cutoff lengths, followed by an accurate optimization of the linear parameters using the new method. For some systems, good values of the cutoff lengths can be supplied immediately (for example, in the fluid phase of an electron gas, the cutoff length L_u should be set equal to the Wigner–Seitz radius of the simulation cell), and one can make use of the new optimization method straight away.

The quartic LSF coefficients are stored in a file called `varmin_coeffs.data`. This file is generated in a VMC simulation when the **use_newopt** keyword is set to "T". If a VMC simulation is performed in a directory in which a `varmin_coeffs.data` file is already present, the existing data in the file will be added to; it is therefore possible to refine the quality of quartic-coefficient data by extending a VMC calculation. Note that `varmin_coeffs.data` is an unformatted file; however, a formatting/unformatting utility called FORMAT_VARMIN_COEFFS exists. This utility reads `varmin_coeffs.data` and produces a formatted `varmin_coeffs.data_formatted` file, and *vice versa*. `varmin_coeffs.data` is deleted by the RUNVARMIN script at the end of each optimization calculation, so that one starts afresh at every VMC "coefficient-generation" stage.

The new optimization method can only be used in conjunction with CASINO's new Jastrow factor. The method is not yet implemented for electron–hole systems.

10.24.3 Further developments to the new optimization method

Extensive tests are currently being carried out to establish the importance of locating the global minimum of the LSF and to determine the best optimization strategy. The following developments to the new optimization method will be made in the near future:

1. The default behaviour of the new optimization method will be improved.
2. The ability to use the new optimization method for electron–hole systems will be added.
3. The new optimization method will be given the ability to read in a `config.in` file containing a set of configurations and construct the quartic coefficients for that configuration set.
4. The new optimization method may be combined with the standard variance-minimization algorithm in such a fashion that the new method is used to optimize linear Jastrow parameters while the nonlinear parameters are optimized by the standard method. Each time the nonlinear parameters are adjusted, the quartic coefficients will be constructed and the LSF will be minimized with respect to the linear parameters using the new method.

10.25 Further considerations in electron-hole systems

CASINO has the ability to include positively charged particles of variable mass (holes) in the simulation in addition to electrons. Currently these may only be used in electron-hole phases without an external potential, but the code needs only a few trivial changes for these things to be able to wander around inside real crystals (useful for studying positron problems - contact MDT if you want this to be implemented).

In this section the changes required to the CASINO code and to the basic equations in the presence of holes are discussed. These largely stem from the possibility of having a variable mass ratio between the positively and negatively-charged particles. The basic differences are:

- The diffusion Green's function, Eq. 11, becomes,

$$G_{\rm D}(\mathbf{R} \leftarrow \mathbf{R}', \tau) = \frac{1}{(4\pi D_e \tau)^{3N_e/2}} \exp\left(-\frac{(\mathbf{R}_e - \mathbf{R}'_e - 2\tau D_e \mathbf{V}_e(\mathbf{R}'_e))^2}{4D_e\tau}\right) \times \frac{1}{(4\pi D_h \tau)^{3N_h/2}} \exp\left(-\frac{(\mathbf{R}_h - \mathbf{R}'_h - 2\tau D_h \mathbf{V}_h(\mathbf{R}'_h))^2}{4D_h\tau}\right), \quad (190)$$

  where e and h denote electron and hole quantities, N_e and N_h are the numbers of electrons and holes, the diffusion constants are defined as $D_e = \frac{1}{2m_e}$ and $D_h = \frac{1}{2m_h}$, where m_e and m_h are the electron and hole masses in atomic units (i.e., in units of the mass of the electron).

- When particle i is moved, Eq. 15 becomes,

$$\mathbf{r}_i = \mathbf{r}'_i + \chi + 2D_i\tau \mathbf{v}_i(\mathbf{R}'), \quad (191)$$

  where χ is a 3-dimensional vector of normally distributed numbers with variance $2D_i\tau$ and zero mean.

- The probability of accepting this move, Eq. 19 is then,

$$p_i \simeq \min\left[1, \exp\left[(\mathbf{r}'_i - \mathbf{r}_i + \tau D_i(\mathbf{v}_i(\mathbf{R}') - \mathbf{v}_i(\mathbf{R})) \cdot (\mathbf{v}_i(\mathbf{R}') + \mathbf{v}_i(\mathbf{R}))\right] \frac{\Psi(\mathbf{R})^2}{\Psi(\mathbf{R}')^2}\right]. \quad (192)$$

- The effective time step, Eq. 28, is given by,

$$\tau_{\rm eff}(\alpha, m) = \tau \frac{\sum_i m_i p_i \Delta r^2_{{\rm d},i}}{\sum_i m_i \Delta r^2_{{\rm d},i}} \quad (193)$$

- The drift vector limiting, Eq. 29, takes the form,

$$\tilde{\mathbf{v}}_i = \frac{-1 + \sqrt{1 + 4D_i a |\mathbf{v}_i|^2 \tau}}{2a|\mathbf{v}_i|^2 D_i \tau} \mathbf{v}_i \,. \quad (194)$$

- Separate Jastrow factors must be defined for the electron-electron, hole-hole and electron-hole interactions. The general form of the cusp condition for Coulomb interactions is,

$$\frac{1}{\Psi}\frac{d\Psi}{dr}\bigg|_{r=0} = \frac{2q_i q_j \mu_{ij}}{d \pm 1} \,, \quad (195)$$

  where q_i and q_j are the charges in units of the charge of the electron, $\mu_{ij} = m_i m_j/(m_i + m_j)$ is the reduced mass and d is the dimensionality. The minus sign is used for distinguishable particles (e.g., anti-parallel-spin electrons or electron and holes) and the plus sign for indistinguishable particles (e.g., parallel-spin electrons). When combined with Eq. 96, cusp conditions force $F_{ij} = \sqrt{A_{ij}/2\mu_{ij}}$ for distinguishable paricles, and $F_{ij} = \sqrt{A_{ij}/\mu_{ij}}$ for the indistinguishable case. For 2D (Eq. 97), these become $F_{ij} = (A_{ij}/6\mu_{ij})^{2/3}$ and $F_{ij} = (A_{ij}/2\mu_{ij})^{2/3}$, respectively.

- The kinetic energy term in the local energy is modified to include the mass,

$$K = \sum_{i=1}^{N} K_i = \sum_{i=1}^{N} -\frac{1}{2m_i} \Psi(\mathbf{R})^{-1} \nabla_i^2 \Psi(\mathbf{R}) \,. \quad (196)$$

Similarly,

$$T_i = -\frac{1}{4m_i}\nabla_i^2\left(\ln|\Psi|\right) = -\frac{1}{4m_i}\frac{\nabla_i^2\Psi}{\Psi} + \frac{1}{4m_i}\left(\frac{\nabla_i\Psi}{\Psi}\right)^2 , \tag{197}$$

and for the drift vector $\mathbf{F}_i$,

$$\mathbf{F}_i = \frac{1}{\sqrt{2m_i}}\nabla_i\left(\ln|\Psi|\right) = \frac{1}{\sqrt{2m_i}}\frac{\nabla_i\Psi}{\Psi} . \tag{198}$$

- The time steps within VMC for the electrons and holes are set independently. This can be used to improve the efficiency.

10.26 Pair correlation function calculation for electron/electron-hole systems

For electron/electron-hole systems (i.e. when **etype** is specified), CASINO allows the evaluation of the spherically averaged pair correlation function, defined by

$$g_{\sigma_1\sigma_2}(r) = \frac{\Omega}{4\pi r^2}\frac{\int_\Omega \sum_{i,j}|\Psi(\mathbf{R})|^2\,\delta(|\mathbf{r}_i-\mathbf{r}_j|-r)\,d\mathbf{R}}{N_{\sigma_1}N_{\sigma_2}}, \tag{199}$$

where a label σ denotes the type (electron or hole) and spin of a particle, $\mathbf{r}_i$ is the position and N_{σ_1} is the number of particles of type σ_1, similarly $\mathbf{r}_j$ is the position and N_{σ_2} is the number of particles of type σ_2, the sums on i and j run over all N_{σ_1} and N_{σ_2} particles respectively and Ω denotes the volume of the simulation cell. When $\sigma_1=\sigma_2$, the sum is replaced by $\sum_{i,j\neq i}$.

In practice, $g(r)$ is evaluated during the simulation by accumulating in radial bins that go up to the Wigner-Seitz radius L_{WS}. The quantity that CASINO actually evaluates is

$$g_{\sigma_1\sigma_2}(r_n) = \frac{\Omega}{N_{\sigma_1}N_{\sigma_2}V_n}\left\langle K_n\right\rangle , \tag{200}$$

where r_n is the radial position corresponding to the nth bin, V_n is the volume of the nth bin and $\langle K_n\rangle$ is an average over the simulation of the number of pairs of particles whose distance falls in the nth bin. In practice this means that at each step of the simulation, N_1N_2 records of distances are made ($N_{\sigma_1}(N_{\sigma_1}-1)$ when $\sigma_1=\sigma_2$).

The radial position and volume of the nth bin in 3 dimensions is given by [47]

$$r_n = \Delta\frac{n^3-\frac{3}{2}n^2+n-\frac{1}{4}}{n^2-n+\frac{1}{3}} \tag{201}$$

$$V_n = 4\pi\Delta^3(n^2-n+\frac{1}{3}). \tag{202}$$

Similarly, in 2 dimensions

$$r_n = \Delta\frac{n^2-n+\frac{1}{3}}{n-\frac{1}{2}} \tag{203}$$

$$V_n = 2\pi\Delta^2(n-\frac{1}{2}). \tag{204}$$

The quantity Δ is the width of each bin, given by $\Delta=\frac{L_{\mathrm{WS}}}{n_{\mathrm{bin}}}$, where n_{bin} is the total number of bins.

From Eqs. (199) and (200) it follows that the normalization of $g(r)$ is such that

$$\int_\Omega g_{\sigma_1\sigma_2}(r)4\pi r^2dr \approx \sum_n V_n\,g_{\sigma_1\sigma_2}(r_n) = \Omega\left(1-\frac{1}{N_{\sigma_1}}\delta_{\sigma_1\sigma_2}\right). \tag{205}$$

To tell CASINO to evaluate $g(r)$, set the pair correlation function flag to **.true.** and specify the total number of bins in the `heg.data` file. All data about $g(r)$ for all particle and spin combinations is then output at the end of the simulation in the `vmc.corr` and `dmc.corr` files. The utility *plot_corr* must be used in order to obtain $g(r)$ for the desired pair of particle types and to apply correct normalization to the data.

10.27 Relativistic corrections to atomic energies

Relativistic corrections to the ground state energies of closed shell atoms can be calculated using first-order perturbation theory, which gives results accurate to the order $1/c^2$ where c is the velocity of light. This method works well for atoms of low nuclear charge Z when the relativistic corrections are small, but is not satisfactory when Z is large. In CASINO the relativistic corrections can be calculated for VMC method 1 and DMC, by setting the **relativistic** flag in the input file to 'T'. By default the relativistic corrections are not calculated.

First we consider the mass polarization term ε_1, which accounts for the correction due to the finite mass of the nucleus to order $1/M$, where M is the nuclear mass in amu. This term is given by

$$\varepsilon_1 = \frac{1}{M}\sum_{i<j}\mathbf{v}_i\cdot\mathbf{v}_j, \tag{206}$$

where $\mathbf{v}_i(\mathbf{R})$ is the drift vector of electron i, $\mathbf{v}_i(R) = \Psi(\mathbf{R})^{-1}\nabla_i\Psi(\mathbf{R})$. By default CASINO uses nuclear masses averaged over isotopes which are listed in Table 2. If a nuclear mass for a specific isotope is required, the default setting can be overwritten by the **isotope_mass** keyword in the input file.

The relativistic terms can be written as a sum of the mass-velocity term, Darwin terms and the retardation term. The mass-velocity term ε_2 arises from the relativistic variation of mass with velocity, and is written as

$$\varepsilon_2 = -\frac{1}{8c^2}\sum_i(\nabla_i\cdot\mathbf{v}_i + \mathbf{v}_i^2)^2. \tag{207}$$

This term is proportional to the square of the nonrelativistic kinetic energy. The spread of electronic charge is described by the electron-nucleus and electron-electron Darwin terms $\varepsilon_3 + \varepsilon_4$, which are expressed together as

$$\varepsilon_3 + \varepsilon_4 = \frac{1}{4c^2}\left[\sum_i(\nabla_i\mathbf{v}_i + 2\mathbf{v}_i^2)\right]\times\left[-\sum_j\frac{Z}{r_j} - \sum_{j<k}\frac{1}{|\mathbf{r}_j-\mathbf{r}_k|}\right], \tag{208}$$

with Z as the nuclear charge. The last term, known as the retardation term ε_5, arises from the interaction between spin magnetic moments which are not mutually penetrating. It is given by the expression

$$\varepsilon_5 = -\frac{1}{2c^2}\sum_{i<j}\left[\frac{(\mathbf{r}_{ij}\cdot\mathbf{v}_i)(\mathbf{r}_{ij}\cdot\mathbf{v}_j)}{\mathbf{r}_{ij}^3} + \frac{\mathbf{v}_i\cdot\mathbf{v}_j}{r_{ij}}\right] \tag{209}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$. Calculations for the beryllium atom [46] show that the total relativistic correction to the energy is approximately 0.00239Ha, with the mass-velocity term having the greatest contribution of 0.0145Ha, followed by the Darwin terms of 0.0119Ha.

1 H 1.00794																	2 He 4.00260
3 Li 6.941	4 Be 9.012187										5 B 10.811	6 C 12.0107	7 N 14.00674	8 O 15.9994	9 F 18.99840	10 Ne 20.1797
11 Na 22.98977	12 Mg 24.3050										13 Al 26.98154	14 Si 28.0855	15 P 30.97376	16 S 32.066	17 Cl 35.4527	18 Ar 39.948
19 K 39.0983	20 Ca 40.078	21 Sc 44.95591	22 Ti 47.867	23 V 50.9415	34 Cr 51.9961	25 Mn 54.93805	26 Fe 55.845	27 Co 58.93320	28 Ni 58.6934	29 Cu 63.546	30 Zn 65.39	31 Ga 69.723	32 Ge 72.61	33 As 74.92160	34 Se 78.96	35 Br 79.904	36 Kr 83.80
37 Rb 85.4678	38 Sr 87.62	39 Y 88.90585	40 Zr 91.224	41 Nb 92.90638	42 Mo 95.94	43 Tc 98.0	44 Ru 101.07	45 Rh 102.90550	46 Pd 106.42	47 Ag 107.8682	48 Cd 112.411	49 In 114.818	50 Sn 118.710	51 Sb 121.760	52 Te 127.60	53 I 126.90447	54 Xe 131.29
55 Cs 132.90545	56 Ba 137.327	71 Lu 174.967	72 Hf 178.49	73 Ta 180.9479	74 W 183.84	75 Re 186.207	76 Os 190.23	77 Ir 192.217	78 Pt 195.078	79 Au 196.96655	80 Hg 200.59	81 Tl 204.3833	82 Pb 207.2	83 Bi 208.98038	84 Po 209.0	85 At 210.0	86 Rn 222.0
87 Fr 223.0	88 Ra 226.0	103 Lr 262.0	104 Rf 261.0	105 Db 262.0	106 Sg 263.0	107 Bh 264.0	108 Hs 265.0	109 Mt 268.0	110 Ds	111 Uuu	112 Uub	113 Uut	114 Uuq	115 Uup	116 Uuh	117 Uus	118 Uuo

Lanthanoids	57 La 138.9055	58 Ce 140.116	59 Pr 140.90765	60 Nd 144.24	61 Pm 145.0	62 Sm 150.36	63 Eu 151.964	64 Gd 157.25	65 Tb 158.92534	66 Dy 162.50	67 Ho 164.93032	68 Er 167.26	69 Tm 168.93421	70 Yb 173.04
Actinoids	89 Ac 227.0	90 Th 232.0381	91 Pa 231.03588	92 U 238.0289	93 Np 237.0	94 Pu 244.0	95 Am 243.0	96 Cm 247.0	97 Bk 247.0	98 Cf 251.0	99 Es 252.0	100 Fm 257.0	101 Md 258.0	102 No 259.0

Table 2: The default nuclear masses used in CASINO.

11 Making movies with CASINO

11.1 How to make movies

In the input file the following block entry controls the movie making process:

```
# MOVIES
makemovie         : T                              #*! Make movie (Boolean)
movieplot         : 1                              #*! Plot every * moves (Inte
movienode         : 0                              #*! Node to plot (Integer)
moviecells        : F                              #*! Plot nn cells (Boolean)
```

Set the keyword **makemovie** to **T** to enable the movie making facility. You could set **movieplot** to an integer greater than 1 so that the particle positions are only written out every **movieplot** moves. The node which plots the particle positions are chosen by the keyword **movienode**, which has to be zero or a positive integer less than the total number of nodes. If **moviecells** is set to **F** then the unit cell will be plotted, if set to **T** then nearest-neighbour cells in the (x, y)-plane will also be written.

Type *runvmc* and an output file `movie.out` will be produced. This is a file in the standard `xyz` molecular format. And example `movie.out` file will look like:

```
           4
 Input geometry
C     0.000000     0.000000     0.000000     5.000000
H    -1.407651    -1.138185     0.054434    -1.000000
H     0.894354     0.554315     1.263301    -1.000000
H     0.528074     1.081535    -0.755823    -1.000000
           4
 Input geometry
C     0.000000     0.000000     0.000000     5.000000
H    -1.407651    -1.138185     0.054434    -1.000000
H     0.894354     0.554315     1.263301    -1.000000
H     0.528074     1.081535    -0.755823    -1.000000
(etc.)
```

Line 1 indicates the total number n of ions and particles. In this case we have 4 particles. Line 2 is a line of comment. The following n lines consist of 5 columns. Column 1 specifies the type of particle (`H` = electron, `O` = hole and `C` = other atoms - of course this labelling is silly for our purposes, but we didn't invent the format). Columns 2, 3, 4 are the x, y, z coordinates of the particle. Column 5 specifies the charge of the particle. This information is then repeated for the number of times specified in the CASINO input file. For example if we run a vmc calculation for **nequil** = 2, **nblock** = 1, **nmoves** = 2 and **movieplot** = 1, then 5 sets of geometry will be created in the `movie.out` file.

11.2 Visualisation

Having generated the `movie.out` file we are now able to visualise the results with VMD or Jmol.

11.2.1 VMD

VMD (Visual Molecular Dynamics) is a molecular visualization program. It supports computers running MacOS-X, Unix, or Windows, is distributed free of charge, and includes source code. VMD can be downloaded from the following website: http://www.ks.uiuc.edu/Research/vmd/

- Type *vmd*; a 'VMD console' and a 'VMD Display' window will appear.
- In the 'VMD console' window type *menu main on*. An extra 'VMD Main' menu bar will appear.
- On the 'VMD Main' menu bar, click on **File** → **New Molecule**. A 'Molecule File Browser' will appear.

- In the 'Molecule File Browser', browse for the file `movie.out` and choose the file type to be **xyz**. Click **Load** to open the file.
- On 'VMD Main', click on **Graphics** → **Representations**. A 'Graphical Representations' menu bar will pop up. Choose **CPK** as the drawing method, the bond resolution to be **1** and the sphere resolution to be **15**. Click **Apply**.
- On 'VMD Main', click on **Extensions** → **vmdmovie**. (For version 1.8.3, click on **Extensions** → **Visualisation** → **Movie Maker**.) A 'VMD Movie Generator' will pop up.
- In **Movie Settings** choose **Trajectory**. The movie can be saved in the AVI or MPEG format. Choose by clicking on **Format** and tick the preferred format. Then check whether the the name of the temporary directory suggested is right (this is where the RGB files are created). Note that this directory should be free of RGB files belonging to other users. If this has to be changed then click on the **Set working directory** button and browse for the directory.
- Type in the name of the movie in the box provided. Click on the **Make Movie** button.
- The movie will be displayed in the **Open GL Display** screen. The .mpg or .avi movie file will be produced in the working directory being specified. They have to be viewed with other viewers, for example *mpeg_play* for .mpg files.

  For an example movie made with VMD (a CASINO VMC simulation of cyclohexane) see `www.tcm.phy.cam.ac.uk/~mdt26/downloads/cyclohexane2.mpg` .

11.2.2 Jmol

Jmol is a free, open source molecule viewer. It supports computers running Windows, Mac OS X, and Linux/Unix systems. Jmol can be downloaded from the following website: http://jmol.sourceforge.net/

- Type 'jmol'. Click on **File**→ **Open**. Browse for the file `movie.out` and click **Open**.
- Click on **Display** and untick the box for **Bonds**.
- Click on **Extras** → **Animate**. An animation tool bar will appear. To start the movie click on the 'play' symbol.

12 Using CASINO with the CRYSTAL program

[see `CASINO/utils/wfn_converters/crystal_9x` and `crystal_03`]

12.1 The CRYSTAL program

CRYSTAL is a commercially available quantum–mechanical electronic structure package which is able to calculate the electronic structure both of molecules and of systems with periodic boundary conditions in 1, 2 or 3 dimensions (polymers, slabs and crystals) using either Hartree-Fock or density functional theory. The latest version of the program was written by Roberto Dovesi, Vic Saunders, Carla Roetti, Roberto Orlando, Nic Harrison, Claudio Zicovich-Wilson, Klaus Doll, Bartolomeo Civalleri, Ian Bush, Philippe D'Arco and Miquel Llunell. See `www.tcm.phy.cam.ac.uk/~mdt26/crystal.html` and the various links thereon for more information.

CASINO supposedly supports the official releases CRYSTAL95, CRYSTAL98 and CRYSTAL03 (but not 88 or 92). Later versions of CRYSTAL03 contain a routine which will write out a CASINO `gwfn.data` file firectly. If you want to use CRYSTAL95 or CRYSTAL98, then you must use a separate utility *crystaltoqmc* which transforms the output of CRYSTAL into a gwfn.data file.

12.2 Generating `gwfn.data` files with CRYSTAL95/98 and *crystaltoqmc*

The utility *crystaltoqmc* (written by MDT) together with the driver script *crysgen* is used to convert the output of CRYSTAL98 into a gwfn.data file readable by CASINO. I will assume you know how to run the CRYSTAL program—I am aware this is a big assumption. You need to run the program using the *run* script, which lives in `CASINO/wfn_converters/crystal9x/run_script`, using the -qmc flag as an argument (this should be set up automatically for you during utilities compilation). The publicly available *run* script on MDT's CRYSTAL web site does not contain this flag, so you need to use the script included with CASINO instead. You will need to change some environment variable definitions in the *run* and *crysgen* scripts to get them to work properly on your system.

If you want to use your own copy of CRYSTAL95 or CRYSTAL98 to generate QMC wave functions, you will need to make some minor modifications to the source code (see the accompanying `CASINO/wfn_converters/crystal_to_qmc/README_CHANGES` file).

(NOTE : 4/2003 - the above changes involve turning off the use of symmetry in k space - for some *metallic* calculations this seems to cause a problem (Gilat net?) unless you also manually turn off all symmetry in the input file with the keyword SYMMREMO. This might be fixed one day.).

CRYSTAL will produce three binary files in the scratch directory you define in the run script—namely `fort.12` (basis set, geometry, common variables), `fort.10` (orbital coefficients) and `fort.30` (eigenvalues). These files would normally be deleted at the end of a CRYSTAL run. Including the -qmc flag as an argument to the *run* script means these files will be grabbed and renamed as `silicon.f12`, `silicon.f10` and `silicon.f30` (or whatever). They will be kept in the scratch directory since they can become very large.

Sit in the directory where these three files live and type 'crysgen'. This will run the *crystaltoqmc* program, which will ask you some questions then generate the `gwfn.data` file.

crystaltoqmc will ask you for the size of the Monkhorst-Pack k-point net in the CRYSTAL calculation, and the desired size of supercell in the QMC calculation. These need not be the same (if not, you are 'plucking' to use the local vernacular) but the former must be divisible by the latter. For example, 12×12×12 MP net in CRYSTAL will allow you to generate `gwfn.data` for 1×1×1,2×2×2,3×3×3,4×4×4 and 6×6×6 supercell cases. Note that it is desirable to carry out the calculation on a higher density **k**-point grid than you actually need so that the orbital coefficients are calculated accurately.

Note that for polymer and slab calculations the last one and two numbers in the MP net and supercell specifications should be 1 to reflect the fact that the system is not periodic in those dimensions. For example, polymer (12×1×1 ⟶ 1×1×1, 2×1×1, 3×1×1, 4×1×1, 6×1×1) slab (12×1×1 ⟶ 1×1×1, 2×2×1, 3×3×1, 4×4×1, 6×6×1) For molecular calculations, just imagine you have a 1×1×1 MP net.

So, to summarize, to produce `gwfn.data` from the crystal input file `dna`:

`% run -qmc dna`

the -qmc flag invokes generation of relevant QMC files in temp

`% cd /temp/mdt`

or whatever the temp directory is called in your CRYSTAL run script

`% crysgen`

then answer the questions

`% mv gwfn.data ~ ; rm dna.*`

then run CASINO

12.3 Generating gwfn.data files with CRYSTAL03

The `gwfn.data` file can be generated directly using the CRYSTAL03 properties program. Note that this facility only exists in the official version of CRYSTAL in binaries produced after December 2003, and that some early versions of CRYSTAL03 had a broken pseudopotential evaluator (which should now be fixed).

The CRYSTAL *run* script included with CASINO will actually force the production of `gwfn.data` automatically if you invoke it with the `-qmc` flag when running the calculations i.e. all you need to is type `run -qmc input_filename`. If the calculation is periodic, the script will ask you how many different supercell sizes N you wish to generate, and then to input N sets of integer triplets indicating the supercell sizes (these must be a subset of the MP shrinking factors in the CRYSTAL input file). For example

% `run -qmc h`

Number of different QMC supercell sizes to calculate? (Maximum 5)

% `1`

Size of cell 1? (e.g. 2 2 2)

% `2 2 2`

Put the script in the background with 'Ctrl-Z' then 'bg'.

If you want to do this by hand, rather than letting the *run* script do it for you, then the relevant part of the CRYSTAL properties input file looks like this (for periodic systems):

```
QMC
2                ! want to generate 2 supercell sizes i.e.
2 2 2            ! a 2x2x2 one
3 3 3            ! and a 3x3x3 one
END
```

For molecules, only the keyword 'QMC' is required with no additional input.

13 Using CASINO with the Gaussian94/98/03 programs

[see CASINO/utils/wfn_converters/gaussian9x-03]
Gaussian is an extremely large and widely-used commercially available quantum chemistry package. More details are available from www.gaussian.com.

gaussiantoqmc is a utility to read the wave function from the output of a Gaussian94/98/03 (G94/G98/G03) calculation and output it in a form compatible with CASINO. *gaussiantoqmc* was originally written by Andrew Porter (2000). Note that there is an associated utility called `egaussian` which extracts the SCF energy (and components) from a Gaussian output file.

The *gaussiantoqmc* code *requires* the existence of a formatted checkpoint file (produced by putting FormCheck=(MO,Basis) in the route section of the Gaussian job file for G94/G98, produced automatically by G03). It expects this file to have a '.Fchk' suffix. The output file of the Gaussian job is also *required*. It is assumed that this has a '.out' suffix. If the original Gaussian job file is present then it will be appended to the end of the QMC input file (as is the Gaussian output file).

13.1 How to use *gaussiantoqmc*

If you have a Gaussian job file called dna (say) and run it to produce dna.out and Test.FChk then you must:

> `> mv Test.FChk dna.Fchk`
>
> run *gaussiantoqmc* ... and follow the prompts
>
> `> mv dna.qmc gwfn.data`
>
> run CASINO

The code should automatically detect what sort of Gaussian job it is and give you the opportunity to construct an excited-state wave function if applicable.

It can deal with the following sorts of calculation:

- HF and DFT ground states, open and closed shell.
- CIS excited states, open and closed shell.
- CASSCF states. Getting Gaussian to output these can be problematic for large calculations. It is possible that *gaussiantoqmc* will get confused if you use some combination of IOps other than those described in Section 13.2.5.
- Time-dependent HF (TD-HF) or DFT (TD-DFT) excited states.

If the user chooses to output a CIS or TD-DFT wave function, they are given the option of resumming it. The wave function must also be resummed if the user wishes to analyse its composition. As distributed, *gaussiantoqmc* will not do this analysis but if you wish to switch this option on then the flag `analyse_cis` in "cis_data.f90" must be set to `.true.` and *gaussiantoqmc* recompiled.

With this flag set the code evaluates the percentage contribution of each single excitation to the CIS expansion. Degenerate virtual and occupied orbitals are identified and their contributions summed. The final output takes the form of files called "from$i$_j.dat" where i–j indicates a range of degenerate occupied orbitals (if $i = j$ then i is a non-degenerate orbital). These files detail all excitations out of the specified orbitals along with a percentage giving their contribution to the CIS expansion as a whole. The sum of the percentages (final column) from each of the "from$i$_j.dat" should be 100 if everything is working OK.

13.2 Other features of *gaussiantoqmc*

The code also contains some crude normalization and plotting routines that are really just debugging aids. By setting the flag `test=.true.` in `gaussiantoqmc.f90` and recompiling, the user is given the option of plotting and testing the normalization of individual molecular orbitals. The axis along which the plotting is done is set in `wfn_construct.f90` and this must be hacked if the user wishes to change things.

13.2.1 Getting Gaussian to do what you want

In principle, Gaussian can do an awful lot of things. In fact, some of these things seem to require magical incantations. These will be described in this section, broken down into the different calculations to which they apply. The comments on G98 refer to revision A9 and may depend on which version is used.

13.2.2 General bits and pieces

Some points to note:

- Both G94 and G98 appear to have have formatting errors when printing out the Gaussians used in the ECP expansion—large exponent values are replaced with stars. This does not affect the subsequent calculation.
- G94's ECP (pseudopotential) package will not accept expansions containing more than 13 Gaussians per angular-momentum channel.
- G98's ECP integral package crashes when one attempts to do a large (both in terms of basis and ECP expansion) calculation with symmetry switched on. The solution to this is to switch symmetry off using "Nosymm".
- One cannot use basis functions containing g and higher basis functions with a pseudopotential in G94.
- Obviously, *gaussiantoqmc* needs the molecular orbitals (MOs) produced by the calculation. It gets these from the formatted checkpoint file which is produced by putting "Formcheck=(Basis,MO)" in the route section of the Gaussian job. Alternatively, it may be obtained from the binary checkpoint file (.chk) using the *formchk* utility— see the Gaussian manual.
- Many (but not all) of the IOp's mentioned here are described on Gaussian's website at:

  http://www.gaussian.com/iops.htm

13.2.3 CIS

- **Performing a "HF test" for an excited state:** It is possible to get Gaussian to output a breakdown of the energy of a CIS excited state which may be compared with the results of a determinant-only VMC run. The key to this is the density used to perform the population analysis and other post-SCF calculations. By default, Gaussian uses the density produced by the original SCF run. To get the kinetic, nuclear-nuclear potential and electron-nuclear potential energies you must tell it to use the one-particle CI density via "density=RhoCI". You must also specify the excited state that you are interested in via "Root=N" in the CIS options. With all of this done properly, Gaussian produces some output like:

  ```
  N-N= 6.9546274D+00 E-N=-2.3781323D+01 KE= 3.3771062D+00
  ```

  (units of Ha) which is hidden in the density analysis right at the end of the output.
- Some trouble has been encountered with the "Add=N" option to CIS (which reads converged excited states from the checkpoint file and then calculates N more). The IOp alternative which does work is IOp(9/49=2) (use guess vectors from the checkpoint file) combined with IOp(9/39=N) (make N additional guesses to those present).

- Using the "50-50" option to CIS to calculate singlet and triplet excitations simultaneously can cause problems. It appears best to do the singlet ("singlets") and triplet ("triplets") calculations separately.

- For QMC, we want the complete CIS expansion. Gaussian may be persuaded to output all excited states with coefficients $> 10^{-N}$ by using IOp(9/40=N). Typically $N = 5$ is good enough. *gaussiantoqmc* outputs the sum of the square of the coefficients so that the user can see how complete the wave function is. (Standard Gaussian output has the coefficients normalized so that their sum of squares for a complete expansion should be unity.)

13.2.4 CISD

Although *gaussiantoqmc* cannot read a CISD wave function it might be worth mentioning that IOp(9/6)=N is equivalent to MAXCYCLE=N for such a calculation.

13.2.5 CASSCF

- As described in Foresman and Frisch's book [34], getting CASSCF to converge for a singlet state is difficult. The following procedure normally works:

  1. Run a ROHF calculation for the lowest triplet state of the system and save the checkpoint file.
  2. Run CASSCF for the second triplet state ("Nroot=2") taking the initial guess from the checkpoint file. (Gaussian calculates the first and second triplets but converges on the second.)
  3. Run CASSCF for the first triplet ("Nroot=1") taking the initial guess from the checkpoint file.
  4. Run CASSCF for the singlet excited state ("Nroot=2") taking the initial guess from the triplet checkpoint file.
  5. Finally, run CASSCF for the singlet ground state ("Nroot=1").

- By default, Gaussian uses spin configurations (combinations of Slater determinants) in a CASSCF calculation. It is best to converge the CASSCF state that you want using this option. However, for input to the QMC code, the wave function must be in terms of Slater determinants. In principle, the "SlaterDet" option to CASSCF will do this but I never succeeded in getting it to work. Instead, specifying IOp(4/21=10) does the trick, as does IOp(4/46=3).

- For large CASSCF calculations (which in Silane equates to an active space of more than six electrons and eight orbitals) the Cray T3E (Turing) at Manchester does not have enough memory per node and simply cannot be used.

- For large CASSCF calculations on the alpha cluster Columbus, the diagonalization method must be changed by specifying IOp(5/51=1).

- In large CASSCF calculations where very many determinants are involved, Gaussian currently only prints the first fifty determinants in the expansion (those with the largest coefficients). The significance of the truncation may be judged by looking at the sum of the squares of the coefficients that *gaussiantoqmc* outputs when it reads the wave function. Unfortunately, this truncation prevents a full "HF test" (*i.e.,* running the wave function in VMC without a Jastrow factor and checking that the result agrees with that of Gaussian) but the energy returned by such a test should be *above* that reported by Gaussian.

- As well as this truncation to fifty determinants, Gaussian has a formatting error which means that if the index number of a determinant is greater than 99,999 then it is replaced by stars and is thus useless. *gaussiantoqmc* deals with this by simply throwing away such configurations which further truncates the expansion.

- Gaussian switches to direct mode for large CASSCF calculations and in doing so automatically stops printing the definitions of the Slater determinants used in the calculation. In order to reconstruct the wave function we do of course need to know what the Slater determinants are. Gaussian may be persuaded to print them by using IOp(4/46=3) IOp(4/21=10) IOp(4/21=100). The first two of these both tell Gaussian to use Slater determinants (I specified both to be on the safe side) and the last one in theory tells it to "just print the configurations" although in actual fact it still proceeds to do the calculation as well.
- Restarting a CASSCF calculation from a previously converged run fails when symmetry is switched on. Use "Nosymm" to avoid this problem.

13.2.6 TD-HF and TD-DFT

As far as converting the resulting wave function for use in CASINO is concerned, these two methods are no different to CIS apart from the issue of normalization. In CIS, the default output is normalized such that the sum of the squares of the coefficients is equal to unity. In a TD-HF or TD-DFT calculation (which involves solving a non-Hermitian eigenvalue problem) a different scheme is used which essentially means that the sum of the squares of the coefficients is arbitrary.

It should also be noted that Gaussian cannot do gradients within TD-DFT yet and so cannot relax excited states.

13.3 Summary of routines used in *gaussiantoqmc*

Routine	Purpose
analyse_cis_state	Break the selected CIS/TD-DFT state down into excitations from each distinct occ. MO
awk_like	Module and subroutines to give AWK-like functionality. Used in parsing the Gaussian output files
cas_wfn	Brings each of the CAS configurations into maximum coincidence with the reference configuration and (optionally) calls `resum_cas`
cas_write	Outputs the CAS wave function (*i.e.,* the determinant expansion) to the gwfn.data file
cis_data	MODULE—holds data defining the CIS and other multi-determinant wave functions
con_coeffs	Multiplies the common part of shell normalization factors into the contraction coefficients and adjusts their storage for improved accessibility
fatal	Echoes a string and then kills the program
g94_wave function	MODULE—holds the data about the type of Gaussian run as well as the data defining the MOs *etc.*
gaussiantoqmc	Main driver unit
g_d_type	Evaluates a primitive d-type Gaussian basis function at a specified location in 3D space
g_s_type	Evaluates a primitive s-type Gaussian basis function at a specified location in 3D space
get_gauss_version	Reads the Gaussian output file and identifies whether it is from Gaussian 94 or Gaussian 98
integ_params	MODULE—holds parameters defining granularity of plotting and integration grids as well as which MO to plot/test
max_coincidence	Brings a CIS configuration into maximum coincidence with a specified 'reference' determinant
normalisation_check	Tests the normalization of a specified MO
normalise_ci	Normalizes the CIS expansion. Not necessary for QMC but keeps things tidy and output gives an idea of how complete the expansion is
numsrt	Sort an array into descending (numeric) order and (optionally) keep track of reordering
numsrt_2way	As for `numsrt` but has additional argument to specify ascending or descending order
numsrt_signchange	As for `numsrt` but returns an associated sign change given by multiplying by -1 for each exchange
pack_evcoeffs	Stores the alpha and beta eigenvector coefficients separately and multiply in remaining normalization factors (which differ between d_{xx},$d_{x^2-y^2}$ *etc.*)
paramfile	MODULE—contains `define_pi` and also defines conversion factors for Ha to eV and Bohr to Angstrom
psi	Evaluates an MO of a given spin at a specified point in space

Routine	Purpose
qmc_write	Writes the "gwfn.data" file
re_sum	Resums a CIS/TD-DFT wave function
read_G9xout	Reads the output file produced by the Gaussian job. Gets the nuclear-nuclear potential energy, CIS/TD-DFT/CAS expansion (if present) and HF eigenvalues
read_fchk	Reads the formatted checkpoint file produced by the Gaussian job. Gets the MOs *etc.*
rejig	MODULE—contains `numsrt` and hence prototypes it which is necessary because it has an *optional* argument
resum_cas	Partially resums a CAS expansion
set_parameter_values	Sets the value of Pi and related constants
shell_centres	Identifies the positions of the distinct shell centres and store the first shell index corresponding to each
sum_degen_excite	Called by `analyse_cis`. Loops over excitations out of a given range (i–j) of (degenerate) occ. MOs and sums those that correspond to degenerate final (virtual) MOs. Outputs the results to a "from$i$_j.dat" file.
user_control	Calls the major reading routines and asks the user about excited states and resumming
wfn_construct	Plot an MO (debugging option). Called by `wfn_test`
wfn_test	Asks user about plotting and normalization testing. Optionally calls `wfn_construct` and `normalization_check`.

14 Use of localized orbitals and bases in CASINO

14.1 Theoretical background

14.1.1 Introduction

The rate-determining step in practical QMC calculations is the evaluation of the orbitals in the Slater part of the trial wavefunction after electron moves[15]. We explain how the CPU time for evaluating the orbitals can be made essentially independent of system size. This makes it feasible to simulate systems of hundreds of electrons with sufficient accuracy to resolve their optical gaps.

14.1.2 "Standard" QMC

Let us assume that in "standard" QMC the orbitals are HF or DFT eigenfunctions extending over the entire system, and that the basis functions also extend over the entire system, as is the case with a plane-wave basis or even with a Gaussian basis if the Gaussians are not truncated[16].

Let N be the system size (number of electrons). In standard QMC, after each electron move, we must update $O(N)$ orbitals[17] expanded in $O(N)$ basis functions. Hence the time taken for a configuration move of all the electrons scales as $O(N^3)$.

14.1.3 Localized orbitals

It is easy to show that a nonsingular linear transformation of a set of orbitals can only change the normalization of the Slater determinant of those orbitals.

Consider a set of Bloch orbitals for a periodic system. There exist efficient algorithms for computing the transformation from Bloch orbitals to so-called maximally-localized Wannier functions [40, 41]. The transformation from Bloch to Wannier functions is unitary, so the orthogonality of the orbitals is preserved.

Wannier orbitals are spatially localized, so they can be truncated to zero when sufficiently far from their centres. Therefore, when an electron is moved, only a few orbitals need to be evaluated; the others are zero because the electron is outside their truncation radii. So the number of orbitals to be computed is $O(1)$.

Although the Wannier functions are localized, they are not rigorously zero at their truncation radii. Hence their truncation results in small discontinuities in the Slater determinant. These are potentially serious for QMC because they result in the presence of Dirac delta functions in the kinetic energy integrand. The delta functions cannot be sampled, so their contribution to the total energy is lost. However, in practice, the resulting bias is extremely small provided the truncation radii are sufficiently large.

The discontinuities can be avoided if the orbitals are truncated smoothly over spherical shells by multiplying them by polynomials that are equal to one at "inner truncation radii" and zero at "outer truncation radii".

14.1.4 Localized bases

Suppose the basis functions are zero outside fixed radii about their centres. Then the only functions that have to be evaluated when an electron is moved are those with the electron inside their radii. So the number of basis functions to be computed simply depends on the local environment of the electron and is therefore independent of system size.

Gaussian basis functions can be regarded as localized if they are truncated to zero outside a certain radius. This is done by default in CASINO: Gaussian functions $\exp(-ar^2)$ are assumed to be zero when $\exp(-ar^2) < 10^{-G_T}$, where G_T is the `gautol` input parameter. Gaussian basis sets cannot yet be used in conjunction with localized orbitals, however.

[15] The time spent updating the cofactor matrices will in principle dominate in the limit of large system size, but this limit is not reached in practice.

[16] In CASINO, the Gaussian basis functions *are* truncated.

[17] The Slater part of the wavefunction is written as the product of Slater determinants for spin-up and spin-down electrons. After the move of a spin-up electron, the number of orbitals to be evaluated is equal to the number of spin-up electrons; after the move of a spin-down electron, the number to be evaluated is the number of spin-down electrons.

On the other hand, orbitals can be represented numerically using splines on a grid. If the orbitals are localized then the memory requirements are greatly reduced because we only have to store the spline coefficients needed to evaluate the orbital within its truncation radius.

14.1.5 "Linear-scaling" QMC

If the numbers of orbitals and basis functions to be evaluated are both $O(1)$ then the CPU time for a configuration move scales as $O(N)$. This is what is meant by "linear-scaling QMC". Note, however, that the number of configuration moves required to achieve a given error bar scales as $O(N)$; hence, in practice, "linear-scaling" QMC calculations scale as $O(N^2)$ and it is better to refer to them as "quadratic scaling" calculations. This has not however been the practice in the literature.

14.2 Using CASINO to carry out "linear-scaling" QMC calculations

14.2.1 Generation of Bloch orbitals

At present the Bloch orbitals must be represented in a plane-wave basis. A plane-wave DFT or HF code should be used to generate a `pwfn.data` file as though an ordinary QMC calculation with a plane-wave basis were to be carried out.

Note that the following restrictions apply:

1. The simulation cell must be orthorhombic;
2. The same orbitals must be available for spin-up and spin-down electrons;
3. Only one **k**-point may be used: the **Γ**-point.

The first two restrictions may be lifted in a later release.

14.2.2 Transformation to Wannier orbitals

The `xwannier` utility written by R. Q. Hood can be used to generate maximally-localized Wannier functions using the method of Berghold *et al.* [41]. This program requires a `pwfn.data` file along with an `input_wannier` file of format:

```
   4    ! No. init. states included in Wannier transform
.TRUE.  ! Compute Wannier functions
.FALSE. ! Print additional files. If false, cannot restart
 1d-12  ! Tol. of converged Wannier functions. Default is 1d-12
```

1. The first line allows the user to specify the orbitals to be included in the Wannier transformation: these are the first orbital in `pwfn.data` up to the number given. If one is interested in ground-state calculations then all of the orbitals up to the highest-occupied orbital should be transformed. If, on the other hand, one is interested in calculating optical gaps then it is essential that both the highest occupied and lowest unoccupied orbitals are not transformed, since the trial wavefunction for the first excited state is generated by replacing the highest occupied orbital with the lowest unoccupied one.
2. The second line should always be set to `.true.`.
3. It is possible to restart `xwannier` if the third line is set to `.true.`.
4. The fourth line contains the localization tolerance. The default of 10^{-12} is usually adequate.

The output of `xwannier` is a `pwfn.data.wannier` file, of the same format as `pwfn.data`, holding the Wannier orbitals expanded in a plane-wave basis. A second file, called `wannier_centers.dat`, is also produced. This holds the Cartesian coordinates of the Wannier centres.

14.2.3 Representing orbitals with splines

The input for `generate_spline` consists of: a `pwfn.data` file (with the restrictions listed above); an optional `wannier_centers.dat` file; and a `generate.dat` file.

The `pwfn.data` file should usually be the (renamed) `pwfn.data.wannier` generated by `xwannier`. The corresponding `wannier_centers.dat` file should also be supplied. However, it is possible to bypass the Wannier transformation and generate a spline representation of Bloch orbitals. In this case no `wannier_centers.dat` file is required and the `pwfn.data` file with the Bloch orbitals should be used.

The format of the `generate.dat` file is shown below:

```
.false.    ! Plot out the truncated, splined orbitals
2.d0       ! Multiplication factor of the FFT grid
1          ! 1=> spherical cutoff, 2=> cuboidal cutoff
1          ! 1=> Use norm, 2 => Use fixed diameter
0.997d0    ! Cutoff criterion (if set to 1 above)
10.d0      ! Fixed cutoff diameter (au) (if set to 2 above)
0          ! No. of states for which the whole box is to be used
0.05d0     ! Minimum shell width (au)
2          ! Verbosity level
```

1. If the first line is set to `.true.` then plots of the splined orbitals in the x-, y- and z-directions through the Wannier centres (or the origin for Bloch orbitals) are produced.

2. The second line controls the fineness of the real-space spline grid. The `pwfn.data` file holds plane-wave coefficients on a cuboidal array of reciprocal lattice vectors. The orbital is evaluated at a grid of points in real space using an inverse fast Fourier transform. The real-space grid spacing in a particular direction is given by 2π over the length of the cuboid in the corresponding direction in reciprocal space. So the easiest way to increase the fineness of the real-space grid is to enlarge the cuboid by including extra reciprocal lattice vectors with zero coefficients. The parameter on the second line is the factor by which the cuboid is enlarged in each direction; hence, for example, doubling it has the effect of doubling the fineness of the real-space grid in each direction, giving eight times as many grid points.

3. The third line specifies whether the truncation surfaces of the localized orbitals are spherical (1) or cubic (2). Note that smooth truncation can only be applied if the truncation surfaces are spherical.

4. The fourth line allows the user to choose the method for determining the truncation radii. If this line is set to 1 then the truncation radius for each orbital will be chosen so that a certain percentage of the square of the orbital norm lies inside it; if set to 2, then the user specifies a fixed radius for all the orbitals.

5. The fifth line holds the desired fraction of the norm squared to be contained within the inner truncation radius. It is ignored if the previous line is set to 2.

6. The sixth line holds the fixed inner truncation *diameter* in a.u. if the fourth line is set to 2. If the fourth line is set to 1 then the sixth line is used as an initial guess at the inner truncation diameter.

7. The seventh line holds the number of orbitals N_E that are not to be truncated. If $N_E > 0$ then the orbitals that are not truncated must be listed on the next N_E lines. If $N_E < 0$ then no orbitals will be truncated.

8. The eighth[18] line holds the minimum permitted thickness (in a.u.) of the shell region (between the inner and outer truncation radii), for the smooth truncation of localized orbitals.

[18] Assuming $N_E = 0$.

9. The ninth line controls the amount of output, from 1 (minimal) to 4 (for debugging). If it is set to 2 or above then useful information about the kinetic energy of the truncated and untruncated orbitals is provided. The difference between the total kinetic energies of the truncated and untruncated orbitals (occupied appropriately) gives the order of magnitude of the likely biasing due to the discontinuities in the trial wavefunction resulting from abrupt truncation. It should be ensured that this bias is much less than the desired QMC error bar.

The splined, truncated orbitals can be generated independently. Correspondingly, `generate_spline` can be run in parallel if desired. It uses the MPI library.

The output of `generate_spline` is a `swfn.data` file containing the splined orbitals and geometry information. If the verbosity level is set to 3 or above then information about the location of the truncation radii will be placed in `output` files for each processor. If requested, the plots of the orbitals are placed in files with names such as `plotx.003` (containing a plot along the x-direction through the centre of orbital 3).

The `swfn.data` file is unformatted. There exists a utility called `format_spline` that reads in a `swfn.data` file and produces a formatted `swfn.data.formatted` file, and vice versa. This is useful if `swfn.data` is to be transferred from one platform to another. It is also possible to insert a title in the formatted file, if desired.

14.2.4 Running a QMC calculation

The `swfn.data` file can now be used by CASINO. The usual input files (`input`, `jastrow.data` and pseudopotentials) should also be supplied. The `btype` input parameter should be set to 6.

Please note the following:

- The `isperiodic` input parameter may be set to either `.true.` or `.false.`. In the former case, the orbitals are evaluated using the minimum image of the distance to the Wannier centre, and a periodic electron-electron interaction is used. In the latter case, the contents of the simulation cell are treated as a finite system. Although the original orbitals were represented in a periodic plane-wave basis, once the localized orbitals have been constructed, represented by splines and truncated, we may dispense with the periodicity.

- An explicit list of the occupied orbitals must be provided using `statelist_up` and `statelist_down` blocks in the `input` file. For example:

  ```
  %block statelist_up
  1 2 3 4
  %endblock statelist_up
  ```

  means that orbitals 1, 2, 3 and 4 are occupied by spin-up electrons. An excited state could be specified by replacing the 4 with a 5.

  This differs from the way in which excitations are specified for plane-wave and Gaussian orbitals. The contents of the `wavefunction` block are ignored.

- There are two ways of storing the spline coefficients of localized orbitals. They can be stored in "pointer arrays" which are specially allocated for each occupied orbital, or they can be stored in a single array with an index for the orbital.[19] The former method permits the memory requirements to be tailored to the appropriate size for each orbital; however, pointer arrays are relatively slow to look up. The latter method is faster but not so memory-efficient because the size of the array is determined by the size of the largest orbital.

- The `bsmooth` input parameter is used to specify whether localized orbitals with spherical truncation surfaces should be abruptly or smoothly truncated. If it is set to `.false.` then the abrupt truncation occurs at the outer truncation radius.

- For large systems it is advisable to set the `sparse` input parameter to `.true.` in order to exploit the fact that most of the orbital values are zero when updating cofactor matrices.

[19] In fact two arrays are used: one for the truncated orbitals and one for the untruncated orbitals. Since only one or two orbitals are left untruncated, using separate arrays for "large" and "small" orbitals allows a big memory saving to be made.

15 Using CASINO with Blip functions

15.1 Blip functions

Blip functions were devised by Mike Gillan and implemented in CASINO by Dario Alfé [35].

15.2 The blip conversion utility

The blip utility is used to convert `pwfn.data` files generated by a plane-wave DFT package into `bwfn.data` files where the wave function is represented in a basis set of localized 'blip functions' on a grid. This should make the code go faster and scale better with system size than plane-waves, which are horrible for QMC calculations. It can require a lot of memory and disk though.

See `CASINO/utils/wfn_converters/pw_to_blips`.

The quality of the blip expansion can be increased with the input parameter *xmul* (the converter will ask you about this when it is run). This results in an increasing number of blip coefficients, so a larger memory occupancy in CASINO (but the CPU time should be the same).

To test the quality of the blip expansion set *ltest = .t.*' when the converter asks you about it. The converter then samples the wave function, the Laplacian and the gradient at 1000 points in the simulation cell and computes the quantity:

$$\alpha = \frac{\langle BW \| PW \rangle}{\sqrt{\langle BW \| BW \rangle \langle PW \| PW \rangle}}$$

The closer to 1 this quantity is the better the representation. Through increasing *xmul* one should always be able to get alpha as close to one as wanted.

The gain in speed with respect to plane waves should be of the order of $nPW/64$.

15.3 The blipl conversion utility

Localized orbitals (see Section 14) can also be represented in terms of blips. In order to do this we require a `pwfn.data` file, a `wannier_centers.dat` file and a `blip_input` file. In the `blip_input` file we need to specify the multiplication factor which defines the fineness of the blip grid; we suggest using values greater than one. Multiplication factors of at least two may be needed in some cases in order to get accurate VMC energies, but this is less of an issue in a DMC calculation. The second entry in the input file is a flag which, if set to `.true.`, allows some testing of the quality of the blip representation (see Section 15.2 for more details). The third entry is the number of localized states. If this is lower than the total number of states than the remainder will be treated in the usual non-localized way, and the blip grid extends over the whole system for those states. The fourth entry is a starting guess for the radius of the (spherical) localization region, and the fifth is the percentage of the norm of the orbital contained in the localization region. Finally, the sixth entry is the thickness of the shell region over which the orbital is smoothly truncated to zero.

Type *blipl* to run the utility to produce a `blwfn.data` file.

16 Using CASINO with other supported programs

16.1 TURBOMOLE

See the README file in `CASINO/utils/wfn_converters/turbomole`

16.2 CASTEP 2002

See the README file in `CASINO/utils/wfn_converters/castep`

16.3 PWSCF

See the README file in `CASINO/utils/wfn_converters/pwscf`

16.4 ABINIT

See the README file in `CASINO/utils/wfn_converters/abinit`

17 Using CASINO with unsupported programs

If the program in question uses Gaussian, plane-wave or blip basis sets, then write your own converter to write the output of the program in the appropriate `[x]wfn.data` format and send it to us (mdt26 at cam.ac.uk) for inclusion in future releases. It may be a stand alone program, or a routine to be inserted into the electronic structure package in question.

If your program uses some other basis set, then this could be a major project. Please ask.

18 Utilities provided with the CASINO distribution

A large variety of little programs which do useful things are provided in the CASINO/utils directory.

Here is a reasonably current list of them. No other documentation is provided here—there may be some in the appropriate directory somewhere in utils.

- **abinit_to_casino_pp** : Converts pseudopotentials for the ABINIT program into CASINO format.
- **billy** : Shell script for optimizing basis sets and geometrical parameters with CRYSTAL95/98/03. Reasonably vital for developing decent trial wave functions for CASINO with these programs.
- **casinokill** : Utility for killing CASINO cleanly on workstations (sorting out the scratch disk etc..).
- **casino_to_abinit_pp** : Converts CASINO pseudopotentials into the correct format for the ABINIT program.
- **clearup** : Script for cleaning up after CASINO by removing stray output and indicator files.
- **dfit** : Draw a polynomial through a set of energy points and find the minimum (lives in the `billy` distribution but can be used on its own).
- **density_plotter** : Three programs : *denconvert*, *d2rs*, *plot_density*
  The *denconvert* script runs *d2rs* which converts the `density.data` file (density expanded in plane waves) into a real space density on a grid. This can be converted into a data format readable by *gnuplot* using the *plot_density* utility.
- **extrapolate_N** : Extrapolate HEG energies to infinite electron numbers using a chi-squared fit to Ceperley's extrapolation formula [36]. Documentation available in the relevant directory.
- **extrapolate_N_eh** : As above for electron-hole systems.
- **extrapolate_tau** : Program to extrapolate DMC energies to zero stepsize by fitting a polynomial form to the DMC energy as a function of stepsize. Documentation in the relevant `utils` directory.
- **format_configs** : Format an unformatted `config.in` file $\longrightarrow$ config.in_formatted (or vice versa).
- **format_pos** : Format an unformatted `vmc.posin` file $\longrightarrow$ vmc.posin_formatted (or vice versa).
- **format_spline** : Utility for formatting/unformatting `swfn.data` files.
- **format_varmin_coeffs** : Format an unformatted `varmin_coeffs.data` file $\longrightarrow$ varmin_coeffs.data_formatted (or vice versa).
- **generate_spline** : Utility to construct a spline interpolation in 3D.
- **graphit** : Plot energy vs move data from a DMC run (i.e., the numbers in `dmc.hist`) in an *xmgr* plot.
- **help** : Simple script to invoke the CASINO help system.

```
casinohelp <casino_keyword>     : tells you the definition and type of keyword
casinohelp search <text>        : finds <text> in descriptions of all keywords
casinohelp all                  : lists all possible keywords
casinohelp basic                : lists all basic level keywords
casinohelp inter                : lists all intermediate level keywords
casinohelp expert               : lists all expert level keywords
```

- **ion_dist** Program for automatic generation of the **edist_by_ion** block in the `input` file for CASINO . Currently works only for antiferromagnetic Wigner crystals that have been generated by the CRYSTAL program (since the numbers are generated from an analysis of `gwfn.data`).
- **modify_inputs** : Simultaneously modify all input files in subdirectories off the current directory (e.g. CASINO/examples) by adding or deleting keywords or by changing the value of a keyword.
- **nstring** : Generate integer number sequence from a to b step c. Useful for strings required in `jastrow.data` files..
- **plot_corr** : Plot calculated pair correlation function (ee/eh/hh) as calculated by CASINO and dumped in a `vmc.corr` or `dmc.corr` file.
- **plot_jastrow** : Convert functions defined in the old `jasfun.data` Jastrow factor file into a form suitable for plotting by *xmgr* or whatever. With the new `jastrow.data form` this functionality has been subsumed into CASINO itself (see the jastrow_plot input keyword).
- **plot_hist** : General program to read a `vmc.hist`/`dmc.hist`/`dmc.hist2` files and plot selected quantities as a function of move number.
- **plot_spline** : Code to read an `swfn.data` file and plot selected states.
- **plot_vmc_energy** : Simple program to read a `vmc.hist` file and plot the total energy as a function of move number.
- **plot_vmc_hist** : Simple utility to plot any of the records from a `vmc.hist` file as a function of move number.
- **pretty_printer** : *mpr* —pretty printing script for source code listings which saves trees (assuming *a2ps* has been set up correctly for your system).
- **ptm** : Manipulate pseudopotentials on radial grids or as Gaussian expansions.
- **quad_fit** : Program for carrying out a quadratic fit to a set of data, so as to find a local extremum. Can be used e.g. for graphing DMC (or VMC) energies against a parameter on which they depend (e.g Gaussian exponent in orbitals of Wigner crystal).
- **quickblock** : Simple reblocking utility intended for analyzing very large dmc.hist files (since it only looks at one column at a time). Alternative to *reblock*.
- **reblock** : Perform statistical analysis and reblocking of QMC data. Use this!
- **rmstore** : Remove old `vmc.hist` files produced with the -big option from $TMPDIR/STORE on all machines in TCM (Cambridge interest only).
- **simple_qmc_codes** : Some very simple QMC codes freely available on the web for teaching purposes.
- **trimdmc** : Trims `dmc.hist` and `dmc.hist2` files to the end of the last complete block. This is useful when restarting DMC runs that have been halted (e.g. by running out of time on a parallel machine with a batch queue system).
- **ve** : Simple utility to extract energy, standard error and time taken from VMC standard output file (single block runs only).
- **wfn_converters** :

```
- GAUSSIAN9X/03 --> gwfn.data
- CRYSTAL9X/03 --> gwfn.data
- PW --> bwfn.data} (blips)
- PWSCF --> pwfn.data
- TURBOMOLE --> gwfn.data
- ABINIT --> pwfn.data
- CASTEP --> pwfn.data
```

Utilities to read data from GAUSSIAN 9X/03, CRYSTAL9X/03, PWSCF, CASTEP, ABINIT and TURBOMOLE calculations and convert to CASINO `(x)wfn.data` format.

- **xwannier** : Given real PW wave functions in a `pwfn.data` file this finds the Wannier centres and if desired the Wannier functions of the specified orbitals.

19 Appendix 1 : Old Jastrow factor file: jasfun.data

In Easter 2004, the new Jastrow factor `jastrow.data` described in Section 19 was introduced. It was designed to replace an old unpublished form of Jastrow factor `jasfun.data` which had been used in CASINO since 1999. The old Jastrow is now redundant, but CASINO still fully supports it for purposes of backwards compatibility, hence it is documented here.

The jasfun.data file contains details of a Jastrow factor of the form described in Section 10.14.4.

```
JASTROW                                              ! Header must be JASTROW
Silicon atom, s^2p^2 (ground) state                  ! Header not read
Non-linear rij term                                  ! Header not read
A, L0, LC                                            ! Header not read
   0.44 0.0  0.0                                     ! A, L_0, of u_0 and cutoff
Enable optimization                                  ! Header not read
   F                                                 ! A not for optimization
L' (a.u.)                                            ! Header not read
   0.0                                               ! L' of S_1
Homogeneous rij term                                 ! Header not read
No of parameters per spin                            ! Header not read
   5                                                 ! No of powers in S_1
Enable optimization                                  ! Header not read
   T                                                 ! For optimization
Parallel and antiparallel components present?        ! Header not read
   T                                                 ! S_1 same for // and anti//?
Spin parallel and antiparallel components            ! Header not read
  5.625899453763526E-004  1.122777200374748E-003     ! B' of spin 1 and spin 2
 -3.487554343373239E-004 -5.611467605982052E-004     ! alpha_1 of spin 1 and 2
  9.941108935490427E-005  2.579395477618149E-004     ! alpha_2 of spin 1 and 2
 -1.175244704897280E-004 -2.103573951529612E-004     ! alpha_3 of spin 1 and 2
  2.542237859441641E-005  4.327076908591314E-005     ! alpha_4 of spin 1 and 2
No of sets of atom centred functions                 ! Header not read
   1                                                 ! Number of sets of atoms
SET 1                                                ! A set has same S_1-S_5
No of ions in set                                    ! Header not read
   1                                                 ! Number of ions in set 1
Ions in set                                          ! Header not read
   1                                               ! Identifies ions in set 1
No of powers in ri,rirj,ririj(2),rirjrij(2) terms    ! Header not read
   5 0 0 0 0 0                                       ! No. of powers in S_2-S_5
L (a.u.)                                             ! Header not read
   0.0                                               ! L of S_2-S_5
ri dependent terms                                   ! Header not read
Enable optimization                                  ! Header not read
   T                                                 ! For optimization
Enable spin polarization                             ! Header not read
   F                                                 ! S_2 to depend on spin?
Components                                           ! Header not read
  8.773613686917481E-003                             ! B of S_2
 -1.810651282199839E-003                             ! beta_1 of S_2
 -1.410044958605923E-004                             ! beta_2 of S_2
 -7.466197634637714E-004                             ! beta_3 of S_2
 -5.119083682175356E-005                             ! beta_4 of S_2
END SET 1                                            ! Denotes end of set 1
END JASTROW                                          ! Denotes end of file
```

Notes:

1. The file is read by SUBROUTINE READ_JASFUN.

2. If the L0 parameter is set to zero, then it takes on a default value. For periodic cases the default is $0.3L_{\mathrm{WS}}$, where L_{WS} is the shortest distance to the boundary of the Wigner-Seitz cell, so that the Gaussian factor in Eq. 96 is small (1.5×10^{-5}) at $r_{ij} = L_{\mathrm{WS}}$. The default L0 for a finite system (atom or molecule) is just a very large number (1×10^{10}) so the function is not 'cut off' at all. Note that any value given to the LC parameter other than zero will override the L0 setting.

3. LC is a parameter outside of which the u_0 function of Eq. 96 is set to zero. If LC<0 then LC is interpreted as the value of u_0 below which it is set to zero, if LC=0 no cutoff is applied, and if LC>0 then the value given is interpreted as the cutoff radius. LC gives a "hard" cutoff of u_0, whereas L0 is a "soft" cutoff. We normally choose LC to be zero and use L0 to apply a soft cutoff.

4. If the L′ parameter is set to zero, then it takes on a default value. For periodic cases the default is L_{WS} i.e. the shortest distance to the boundary of the Wigner-Seitz cell. For molecular systems, the default is 'the molecular size', currently defined as the maximum interatomic distance times 1.25. For atoms, the default is 2.5.

5. if the L parameter in any of the sets of atom-centred functions is set to zero then it is given a default value. The default is equal to the longest of the nearest neighbour distances for the atoms in the set (CASINO will perform a neighbour analysis to work this out). For single atoms which don't have any neighbours, it is currently given the value of 2.0 .

6. Where functions are spin-dependent the first column (1) is for the spin parallel or spin up case, and the second column (2) is for the antiparallel or spin down case.

7. For the "No of powers in ri,rirj,ririj(2),rirjrij(2) terms", the ririj and rirjrij terms require two integers for the $l4$ and $m4$ of Eq. 101 and the $lm5$ and $nm5$ of Eq. 102. At the moment only the terms in S_3 with $l = m$ have been coded. The extra terms will be included in a later release.

Modifications for jellium slab system

The jellium slab, although homogeneous in the plane of the slab (x and y directions), is inhomogeneous in the z direction. The introduction of the usual two-electron term in the Jastrow factor modifies the density away from the desired z-dependence; a one-electron term, depending only on z, is required to restore the correct form. This term is only available for ETYPE=6, and is introduced by including the following lines in jasfun.data:

```
Enable one-body z-dependent term
 T
Enable optimization
 T
No of Fourier components present
 2
Fourier components: frequencies and coefficients
 0.930842267730309  0.335794237334347
 2.79252680319093  -0.005315599024948
```

The lines belong at the end of the file (just before END JASTROW). The Jastrow factor will then include a term

$$J_z = -\sum_k C(k) sin(kz)$$

20 Appendix 2 : Programming Guide for CASINO

You are not allowed to make modifications to CASINO without explicit written permission from the Cambridge group. Should you be in such a position, please read the following.

20.1 STYLE

CASINO has a FORTRAN90 'style' which should be adhered to when writing code either for the main source or for utilities. This is because we think it is desirable that the package has a relatively homogeneous look and feel. Everybody has their own style. Yours is different and may even be better, but we've decided on one for CASINO and here it is. If you don't write your code like this, the likelihood is that somebody else will reformat it for you, and they will probably accidentally delete a crucial minus sign while correcting your routine, an error which may take two weeks of your life (which could be much better spent doing other things) to track down. You can get most of this just by looking at the code, but let's emphasize the main points:

- Don't use more than 1 module per physical file, as Hitachi compilers won't allow it!
- Capitalization outside of character context: Use upper case letters for keywords whose use is primarily at compile time (statements that delimit program and subprogram boundaries, declaration statements of variables). Use lower case letters for everything else, including the bulk of run-time code. Description of routines to be enclosed in a little box with a description of what it does and an author. Later changes to be documented at the bottom of this box. For example:

```
 SUBROUTINE rubbish
!------------------------------------------------------------------------!
! Routine not to do anything at all.                                     !
!                                                                        !
! MDT 8.2000                                                             !
!                                                                        !
! Changes:                                                               !
! --------                                                               !
! 3/2001 MDT - added capability to write 'Hello'                         !
! 5/2001 MDT - added additional capability to write 'Everyday Spanish'   !
!------------------------------------------------------------------------!
  USE dsp
  USE parallel
  IMPLICIT NONE
  INTEGER i,j,k,some_integer
  REAL(dp) a
  CHARACTER(llength)my_name

  i=0
  do j=1,10
   write(6,*)'Hello'
   do k=1,1000000
    i=i+1
    write(6,*)'Everyday Spanish'
   enddo
  enddo

  etc..

 END SUBROUTINE rubbish
```

- Contents of if, do and case blocks to be indented by one space.

- In general don't put spaces between words, e.g.,

```
if(something_is_true)a=b+c
not
if ( something_is_true )  a  =  b  +  c
```

- Use f90 versions of: .eq..ne. .lt. .le. etc.. i.e., ==, /-, <, <=
- Maximum of 80 characters per line. If you go over this, use & continuation characters at the end of the line *and* at the beginning of the next one.
- Two blank lines between subroutines in a given physical file.
- Error conditions to be handled using the errstop and errwarn routines (in the utilities.f90 module).
- Use 'endif' 'enddo' not 'end if' 'end do'.
- No double colons in simple variable declarations e.g.

```
 INTEGER ialloc
except where required e.g.
INTEGER,INTENT(in) :: n
```

- In lists of declared variables, adhere to the following order:

```
INTENTed dummy arguments first, order: in/inout/out

INTEGER,INTENT(in)
(INTEGER,ITENT(inout) etc..)
(INTEGER,INTENT(out) etc..)
REAL(dp),INTENT(in)
COMPLEX(dp),INTENT(in)
CHARACTER(12),INTENT(in)
LOGICAL,INTENT(in)

followed by things which are not arguments

INTEGER
REAL(dp)
COMPLEX(dp)
CHARACTER(12)
LOGICAL
```

  Within each class, put standard variables on the first line, followed by things with attributes like ALLOCATABLE, PARAMETER, etc., on subsequent lines, in whatever order seems aesthetically pleasing.

  Note the CHARACTER(12), not CHARACTER*12, which is not in the Fortran90 standard.

- Don't use tab characters anywhere in the code.
- All units for reading and writing to be allocated unit numbers using the standard 'call open_units' procedure which you can figure out by looking at other examples in the source.
- Initial letter of comments to be in capitals. Comments to be spelled correctly. Comments to be useful.

```
! This is a legitimate comment

! this is an illegitimate comment                                X

! tihs one is evn more illetigimate as i cant spell.             XX

! allocate variables                                             XXX (really?)
  allocate(a(1))
```

- If you add a module USE statement anywhere, don't forget to change the dependency list in the Makefile.
- module USE statements in alphabetical order, followed by USE ONLY's, in alphabetical order

```
USE a
USE b
USE c
USE a1, ONLY : x
USE b1, ONLY : x
USE c1, ONLY : x
```

- Don't use 'return' at the end of each routine - just use END SUBROUTINE or END FUNCTION etc.

20.2 CONTENT

Important policy statement : Following advice from ABINIT chief developer Xavier Gonze, and in order to reduce the workload on Mike and Neil, the policy of the Cambridge group regarding inclusion of routines written for CASINO is now as follows. Such modifications will *not* be accepted under any circumstances unless the modifications are made to the most recent version of the code (email Mike when commencing the merge so that the 'most recent version' does not change too much during this process).

The reason for adopting this policy is simply that merging complex code written by someone else into a program as large and complex as CASINO is a very time-consuming procedure, and is prone to introducing errors. This process can and has taken months with previous submissions.

Even if modifications are made to the latest version, you may find that they are still not incorporated into the public release. If you want this to happen, there are a variety of ways to lower the energy barrier for the inclusion of your routines. The general theme here is to try to reduce the amount of effort that Mike has to perform to validate and check your code. If your submission includes lots of things from the 'Good things' list, and none at all from the 'Bad things' list, then inclusion is semi-automatic and will normally take place within days. We are aware that it requires a lot of work to do the things on the Good list (so sulk about it) but we remind the reader that we will have to do it if you don't.

20.2.1 Good things

- Well written absolutely standard Fortran90 in the standard CASINO format. Do not use fortran95 (or later) constructs, or non-standard (but useful) extensions like 'flush'.
- The results of extensive testing of your routine with a range of examples, with before and after numbers (including timings).
- A detailed description of whatever your code is supposed to do (I shouldn't have to go through each line of your code at the same detailed level you did, or I might as well have written it myself.).

- A bit of TeX for inclusion in the manual (if the code requires the user to know something over and above what the current manual describes).
- Some example input/output which demonstrate the new capabilities that Mike can just run and look at, without having to invent his own examples.
- Evidence of testing on both single-processor and parallel machines. Remember only the master node should be writing to the output file. Also - CASINO is supposed to compile and work on single-processor machines *without* an installed MPI library. Use the fake `comms_serial.f90` module supplied both in the `utils` directory and with the main code.
- You should not use *any* calls to external libraries apart from MPI. This includes BLAS, LAPACK, NAG etc..
- It uses the standard CASINO facilities for handling errors.
- The routine implements something that we desperately want or is in the TODO list.

20.2.2 Bad things

- We do not sacrifice speed for additional functionality. Think of another way to write it if it slows the code down. It's slow enough as it is.
- CASINO does not like to do things unless they are completely general. The development of general complex electronic structure codes can be set back years by people choosing to implement functionality which works only for their current project. Non-general algorithms tend to get completely thrown away and re-implemented which wastes the time of both parties. It is not usually much harder to write a general program than it is to write a specific one (we understand this is not always true!).
- When writing to the output file, the new routines should not write out lots of weird arrays (whose meaning is understood only by the author) in an incredibly scruffy format full of spelling mistakes. In general, be as beautiful and informative as you can when writing to output, or Mike will just have to make it so (which takes ages). If there are all sorts of special cases which require different output, then so be it - select case and if blocks are very useful in this regard..
- Prior agreement for modification of the code was not obtained. Unsolicited submissions are not accepted and can lead to moody lack of cooperation lasting years.

20.3 PERFORMANCE

CASINO prints a timing analysis at the end of the output file when it has finished a calculation. This information can be useful to the programmer in deciding what optimizations to do to make the program run faster.

The total time is divided into 'User' and 'System' time, the sum of which is printed as 'CPU' time. These timings are broken down according to specific tasks, although by no means all activities are timed - only those thought likely to contribute significantly. For very small systems such as atoms a significant amount of CPU time will remain unaccounted for since the traditional heavyweight tasks are very fast. For medium and large systems, often the calculation of the orbitals and their derivatives will dominate, closely followed by Jastrow factors and Ewald interactions. In systems with pseudopotentials, the non-local integration is very expensive indeed (since it calls the orbital evaluation routine a lot) and this expense will increase the larger the non-local cutoff radius (lower nlcutofftol in the input to see the effect of this).

The 'User' time is time spent executing the instructions that make up the user code, including e.g. calls to subroutine libraries linked to the code.

The 'System' time is time spent in *kernel* mode, processing I/O requests, for example, or other situations which require the intervention of the operating system.

Together these comprise the total CPU time. Note that the User time is usually significantly greater than the System time. Indeed, a high ratio of system to user CPU time may indicate a problem. Exceptions such as page faults, misaligned memory references, and floating-point exceptions consume a large amount of system time. Note that time spent doing things like waiting for input from the terminal, seeking on a disk, or rewinding a tape does not appear in the CPU time; these situations do not require the CPU and thus it can (and generally does) work on something else while waiting.

The elapsed ("wallclock" or "real") time is simply the total real time that passed during the execution of the program. This will typically be greater than the total CPU time due primarily to sharing the CPU with other programs. In addition, programs that perform a large number of I/O operations, require more memory bandwidth than is available on the machine (i.e., the CPU spends significant time waiting for data to arrive from memory), or is paged or swapped will show a relatively lower ratio of CPU time used to elapsed time.

20.4 BUG REPORTS

We cannot of course guarantee that CASINO is free of bugs, and if you find one, please tell us. If you don't know what the problem is yourself, than please include as much detailed information you can about the nature of the error in order to ensure a quick fix.

20.5 REQUESTS FOR NEW FEATURES

We are of course always happy to discuss such requests.

References

[1] P. P. Ewald, Ann. Phys. **64**, 25 (1921). 87

[2] M. P. Tosi, in *Solid State Physics*, Vol. 16, edited by H. Ehrenreich and D. Turnbull (Academic, New York, 1964), p. 1. 87

[3] L. Mitáš, E. L. Shirley and D. M. Ceperley, J. Chem. Phys. **95**, 3467 (1991). 85

[4] P. J. Reynolds, D. M. Ceperley, B. J. Alder and W. A. Lester, Jr., J. Chem. Phys. **77**, 5593 (1982). 66, 68

[5] C. J. Umrigar, M. P. Nightingale and K. J. Runge, J. Chem. Phys. **99**, 2865 (1993). 65, 68, 69, 72, 73

[6] C. J. Everett and E. D. Cashwell, *A Third Monte Carlo Sampler*, Los Alamos Technical Report No. LA-9721-MS (1983). 72

[7] C. J. Umrigar and C. Filippi, unpublished. 65, 68

[8] M. F. Depasquale, S. M. Rothstein and J. Vrbik, J. Chem. Phys. **89**, 3629 (1988). 69

[9] H. Flyvbjerg and H. G. Petersen, J. Chem. Phys. **91**, 461 (1989). 92

[10] L. M. Fraser, W. M. C. Foulkes, G. Rajagopal, R. J. Needs, S. D. Kenny and A .J. Williamson, Phys. Rev. B **53**, 1814 (1996). 89

[11] A. J. Williamson, G. Rajagopal, R. J. Needs, L. M. Fraser, W. M. C. Foulkes, Y. Wang and M.-Y. Chou, Phys. Rev. B (Rapid Communications) **55**, 4851 (1997). 89

[12] P. R. C. Kent, R. Q. Hood, A. J. Williamson, R. J. Needs, W. M. C. Foulkes and G. Rajagopal, Phys. Rev. B **59**, 1917 (1999). 74, 89

[13] C. J. Umrigar, K. G. Wilson and J. W. Wilkins, Phys. Rev. Lett. **60**, 1719 (1988).

[14] P. R. C. Kent, R. J. Needs and G. Rajagopal, Phys. Rev. B **59**, 12344 (1999). 95

[15] C. Filippi and C. J. Umrigar, J. Chem. Phys. **105**, 213 (1996). 95

[16] S. Manten and A. Lüchow, J. Chem. Phys. **115**, 5362 (2001). 76

[17] K. E. Schmidt and J. W. Moskowitz, J. Chem. Phys. **93**, 4172 (1990). 95

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. M. Teller and E. Teller, J. Chem. Phys. **21**, 1087 (1953). 62

[19] S. Fahy, X. W. Wang and S. G. Louie, Phys. Rev. B **42**, 3503 (1990). 62, 85

[20] D. M. Ceperley and M. H. Kalos, in *Monte Carlo Methods in Statistical Physics*, edited by K. Binder (Springer, Berlin 1979); K. E. Schmidt and M. H. Kalos, in *Monte Carlo Methods in Statistical Physics II*, edited K. Binder (Springer, Berlin 1984).

[21] M. Dewing, J. Chem. Phys. **113**, 5123 (2000). 63

[22] B. L. Hammond, W. A. Lester, Jr. and P. J. Reynolds, *Monte Carlo methods in ab initio quantum Chemistry*, (World Scientific, Singapore, 1994). 64, 73

[23] W. M. C. Foulkes, L. Mitas, R. J. Needs and G. Rajagopal, Rev. Mod. Phys. **73**, 33 (2001). 62, 63, 64, 67

[24] V. R. Saunders, R. Dovesi, C. Roetti, M. Causà, N. M. Harrison, R. Orlando and C. M. Zicovich-Wilson, CRYSTAL98 User's Manual (University of Torino, Torino, 1998). 4

[25] V. R. Saunders, R. Dovesi, C. Roetti, R. Orlando, C. M. Zicovich-Wilson, N. M. Harrison, K. Doll, B. Civalleri, I. Bush, Ph. D'Arco, M. Llunell, CRYSTAL2003 User's Manual (University of Torino, Torino, 2003). 4

[26] Gaussian 98 (Revision A.7), M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. G. Johnson, W. Chen, M. W. Wong, J. L. Andres, M. Head-Gordon, E. S. Replogle and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 1998. 4

[27] Gaussian 03, Revision B.03, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 2003. 4

[28] M. D. Segall, P. L. D. Lindan, M. J. Probert, C. J. Pickard, P. J. Hasnip, S. J. Clark and M. C. Payne, J. Phys.: Cond. Matt. **14**(11) pp.2717-2743 (2002) 4, 8

[29] *First-principles computation of material properties: the ABINIT software project*, X. Gonze, J.-M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.-M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, M. Torrent, A. Roy, M. Mikami,Ph. Ghosez, J.-Y. Raty, D.C. Allan. Computational Materials Science **25**, 478-492 (2002). 4, 8

[30] S. Baroni, A. Dal Corso, S. de Gironcoli, and P. Giannozzi, `http://www.pwscf.org` 4, 8

[31] W. Müller, J. Flesch and W. Meyer, J. Chem. Phys. **80**, 3297 (1984). 86

[32] E. L. Shirley and R. M. Martin, Phys. Rev. B **47**, 15413 (1993). 85, 86

[33] D. M. Ceperley, M. H. Kalos and J. L. Lebowitz, *Computer simulation of properties of a polymer chain*, Macromolecules **14**, 1472 (1981). 66

[34] J. B. Foresman and M. J. Frisch, *Exploring Chemistry with Electronic Structure Methods*, Gaussian Inc., Pittsburgh, PA, 2nd edition (1996). 108

[35] D. Alfé and M. J. Gillan, Phys. Rev. Lett. **70**, 161101 (2004). 116

[36] D. M. Ceperley and B. J. Alder, Phys. Rev. B. **45**, 566 (1980). 118

[37] V. R. Saunders, C. Freyria-Fava, R. Dovesi, L. Salasco and C. Roetti, Mol. Phys. **77**, 629 (1992). 87

[38] See e.g.D. E. Parry, Surf. Sci. **49**, 433 (1975) and erratum, Surf. Sci. **54**, 195 (1976). 88

[39] J. E. Dennis, D. M. Gay and R. E. Welsch, *Algorithm 573: NL2SOL—An Adaptive Nonlinear Least-Squares Algorithm*, ACM Transactions on Mathematical Software **7**, 369 (1981). 94

[40] N. Marzari and D. Vanderbilt, *Maximally localized generalized Wannier functions for composite energy bands*, Phys. Rev. B **56**, 12847 (1997). 112

[41] G. Berghold, C. J. Mundy, A. H. Romero, J. Hutter and M. Parrinello, *General and efficient algorithms for obtaining maximally localized Wannier functions*, Phys. Rev. B **61**, 10040 (2000). 112, 113

[42] T. Kato, Commun. Pure Appl. Math. **10**, 151 (1957). 76

[43] Y. Kwon, D. M. Ceperley, and R. M. Martin, Phys. Rev. B **48**, 12037 (1993).

[44] Y. Kwon, D. M. Ceperley, and R. M. Martin, Phys. Rev. B **58**, 6800 (1998).

[45] M. Holzmann, D. M. Ceperley, C. Pierleoni, and K. Esler, Phys. Rev. E **68**, 046707 (2003).

[46] *Relativistic corrections to atomic energies from quantum Monte Carlo calculations*, S. D. Kenny, G. Rajagopal, and R. J. Needs, Phys. Rev. A **51**, 1898 (1995). 100

[47] L. M. Fraser, PhD thesis, Imperial College of Science, Technology and Medicine, 1995. 99